Adaptive Mesh Refinement法とその応用 : 究極の高解像度計算を目指して(<シリーズ>物性研究者のための計算手法入門)

AUTHOR(S):

山田, 良透; 宮下, 尚

CITATION:

山田, 良透 ...[et al]. Adaptive Mesh Refinement法とその応用 : 究極の高解像度計算を目指して(<シリーズ>物性研究者のための計算手法入門). 物性研究 2001, 77(1): 73-112

ISSUE DATE:

2001-10-20

URL:

http://hdl.handle.net/2433/97076

シリーズ「物性研究者のための計算手法入門」

Adaptive Mesh Refinement 法とその応用*)

― 究極の高解像度計算を目指して ―

京都大学 理学部　山田 良透*1 、宮下 尚 *2

（2001 年 8 月 27 日受理）

概要

高解像度計算法として、我々は Adaptive Mesh Refinement 法のコード開発を行っている。この手法は、制限された計算機リソースの中で高解像度の計算を行うための、非常に強力な道具である。本稿は、この手法について解説することを目的とする。我々はその効率と、厳密解との比較による精度の評価を行った。その結果、従来論文で発表されている手法には、計算精度を制限してしまう場合があり、改良が必要な部分があることがわかったので、その改良について併せて解説する。最後に、最近我々が行っている、流体力学における特異性発生問題の解析について、解説する。本稿の大部分は、手法の解説に当てられる。

1　Adaptive Mesh Refinement 法とは

1.1　Adaptive Mesh Refinement 法との出会い

私 (山田) が初めて Adaptive Mesh Refinement(以下、AMR と略す) 法に出会ったのは、1989 年、カリフォルニア大学 Santa Cruz 校にて行われた超新星の国際会議の席上、当時 LowrenceLivermore 研究所に所属していた、R. Klein の発表であった。計算領域のあちらこちらに小さな高解像度のメッシュを張り、細かい構造まで resolve しながら広領域の計算を行うという、実に画期的な計算方法だった。しかし、残念ながらこの計算は、当時の Cray で 50 時間を要し、たとえコードがあったとしても、とても我々の計算できるものではなかった。この時代にこのようなコードを使って研究を行うことは、スーパーコンピューターが湯水のように使える環境にいる研究者の特権だったかもしれない。また、この当時の論文をみると、AMR 法の作者の一人である P. Colella は、このころ Lawrence Livermore 研究所に所属していた。このことも、Klein にとっては幸運だったといえる。

しかし、現在では廉価なパーソナルコンピューターでも、演算性能は当時のスーパーコンピューター並みにまで向上し、また各分野の中核となる研究所には並列やベクトルのスーパーコンピューターがあって、その分野の研究者ならかなり自由に使うことができる時代となった。今なら、コードさえ手に入れれば誰でも同様の高解像度の計算を行うことができる。

本稿の読者の中には、このような計算方法を初めてみる方もいるだろうし、また高解像度の計算

*1 E-mail: yamada@scphys.kyoto-u.ac.jp
*2 E-mail: himi@scphys.kyoto-u.ac.jp

*) 本稿は、編集部の方から特にお願いして執筆していただいた記事である。

をしたいけれどもこのような複雑なコード開発には手を出せずに躊躇しているかたもいると思う。そこで、AMR 法の概要と、実装のポイント、およびその応用例として二次元の流体特異性に関する問題を紹介する。メッシュ計算を行う場合、現在のスーパーコンピューターで計算可能な総格子数は、計算速度および使用可能なメモリー容量の双方の制限から、高々10^7 である。従って、等間隔でほぼ正方形のメッシュを使う場合、二次元の場合は一方向あたり 3×10^3 程度、三次元の場合一方向あたり 200 程度が限界ということになる。例題として取り上げる、二次元の流体特異性に関する問題では、一次元的な渦層が形成される。従って、AMR 法を適用すると、実質的に一次元方向の格子数の上限が、等間隔格子計算での一次元計算の上限の数分の一まで増やすことができる。例題として紹介する流体の計算では、計算領域の一辺の長さともっとも細かいメッシュの格子間隔との比は、2×10^6 である。同様の事情から、三次元の中に一次元的な渦糸が形成されるような問題でも、同程度の解像度が期待できる。また、三次元の中に二次元の薄い層ができるような問題では、実質的に一方向あたりの格子数の上限が、二次元等間隔メッシュ計算における一方向あたりの格子数の上限に近い値まで増やすことができると推定される。また、計算領域の中で高解像度を必要とする部分が局在している場合、さらに高い解像度を得ることができる。二次元で一方向 10^6 というのは、AMR 法を適用しない場合には総格子数で 10^{12} ということであり、100TB 以上のメモリーを必要とする。三次元で一方向 10^3 の計算を行った場合でも、総格子数で 10^9 であるから数百 GB 程度のメモリーを必要とする計算になる。これは、現在のコンピューターではとうてい実現不可能な容量であり、そのような問題を現在のコンピューターで解こうと思えば、AMR 法が必須である。

なお、本稿は方法の紹介を目的とするものなので、物理的内容よりも技術的な側面に重点を置いて書かれる点は、本稿依頼の趣旨に免じてご容赦いただきたい。

1.2　物理シミュレーションの手順

まず、このような複雑な計算方法の必要性と有効性を示す前に、物理法則のシミュレーションとはいかにして行われるかという点を整理してみることにする。これは、今後シミュレーション分野でプロジェクト的研究をする上で、重要な指針となる。

物理法則のシミュレーションを行う場合、典型的な場合は、以下の手順が踏まれることになる。

(1). 対象のモデル化に基づく、偏微分方程式の導出　この段階では、対象の研究手法を下に、偏微分方程式を記述する。例えば、物理学的な手法で対象をモデル化する場合は、物理学の理論に基づいた方程式が導出される。

(2). 偏微分方程式の、空間・時間の適切な離散化に基づく、計算スキームの導出　この段階では、数値計算の技術が要求される。与えられた連続的な偏微分方程式を、どのように空間的・時間的に離散化するか、また、時間積分などの方法はどのようにして行うかをこの段階で決定する。分解能、解像度が、数値計算結果に与える精度への影響は、この計算スキームが左右することになる。

(3). 計算スキームに基づいて、実際のコードを作成。　(2) で、決定したスキームを実際の FOR-

TRAN や C 等の計算機言語で記述する。

(4). コードの実行 計算結果の収集　作成したコードを実行し、計算結果を取得する。計算の量や質に応じた計算機リソースが要求される。

(5). 計算結果の誤差評価　計算結果の誤差は、要求される精度の範囲内に収まっているかを検証する。

(6). 計算結果の解釈　もともとのモデルの観点から、計算結果を解釈する。例えば、物理学においては、物理学的な観点から、計算結果を捉えなおす。

一般にシミュレーションを行うためには、(2) で示したとおり、空間、時間の離散化が不可欠であり、精度を向上させるためには、その分解能および解像度を上げなくてはならない。しかし、時間分解能を a 倍に向上させると、当然、その計算量も a 倍に増えてしまう。また、空間解像度を b 倍に向上させると、b^d(d は次元数) 倍で計算量及び、要求記憶量が増してしまう。しかも、近年、計算機の高速化に伴い、(2) を出来るだけ自然な形で実装し、導出された偏微分方程式を、そのまま解く事例、及び、要求が多くなっている。例えば、従来であれば、計算量を劇的に減らすために空間の対称性を仮定して次元数を減らすなどの細工を必要とした。しかし、このような手法は一般的なものではなく、多様化していくシミュレーションの要求に答えきれなくなっている。具体的に例をあげると、空間の次元が 3 次元である場合、シミュレーションの解像度を $b = 2$ 倍高めようとすれば、$b^3 = 8$ 倍のメモリーが要求される。時間発展計算が必要な場合は時間解像度も関係するから、計算時間では少なくとも $b^{3+1} = 16$ 倍必要となる。亜音速気体や電磁場、自己重力を含む系などでは、Poisson 方程式を解かなければならない。Poisson 方程式は、通常緩和法で解かれるが、一般に全体の格子数が増えれば、必要な反復回数も増える。従って、解像度 b に対して要求される計算量 b^r のべき r は、次元数 d に対して上の $d+1$ より大きくなる。つまり、空間解像度を 2 倍を高めようと思っても計算量の増加は 16 倍ではすまず、数十倍から 100 倍以上になることもある。つまり、多次元計算で十分な解像度のシミュレーションを行うことは容易なことではないのである。

1.3 Adaptive Mesh Refinement 法の利点

このような問題を解決させる方法として、約 15 年前から米国の研究者が AMR 法という計算法を開発した [2][3]。この計算法は、必要な部分に局所的に細かい格子を用いて離散化を行い、全体は粗い格子で離散化を行うことにより、計算機資源を節約しつつ高解像度の計算を実現しようというものである。粗い格子で離散化した場合は、時間分解能も荒くしても誤差のオーダーは同等程度であるため、この方法を用いると、有効に時間分解能も必要な部分だけ細かくすることが効率よく行える。この方法は、実際に流体現象 [14] やプラズマ現象 [12] における特異性の発生問題、宇宙の構造形成問題 [20, 24] などに応用され、実用的には航空機の機体設計や天気予報 [21] のためのコード開発にも用いられている。

この方法は、一般的な偏微分方程式の時間発展問題を、離散化誤差の大きいところに空間的・時間的に局所的に分解能を向上させることにより、わずかな計算機資源しか消費しないにも関わら

ず、全体的に小さな離散化誤差で計算を行い、計算時間を著しく短縮させることを目的とする。即ち、同程度の離散化誤差の計算を行う場合においては著しい計算時間の短縮、計算の高速化をもたらすことになる。この方法では、離散化誤差の大きな部分を含む長方形領域に、格子間隔の小さな新たな長方形メッシュを張り、格子間隔の大きなメッシュと格子間隔の小さなメッシュの時間発展計算を同時に行う。格子間隔の小さなメッシュを作成する場所は、空間的時間的に限定的であるため、計算機リソースの消費も少なく抑えることができる。更に、計算の途中で格子間隔の大きなメッシュでも離散化誤差が十分に小さいと判断すれば、その領域の格子間隔の小さなメッシュは取り除くことにより、計算機リソースを解放する。この方法ではメッシュの生成消滅、整合性の確保にはオーバーヘッドがあるが、空間的・時間的に限定された範囲で格子間隔の小さなメッシュを生成することによりオーバーヘッドを相殺して余りある効果がある。

1.4　Adaptive Mesh Refinement 法の将来性

物理シミュレーションは、物理学研究のみならず、実用的な応用も広い。ビルを建設することに伴う空気の流れの変化、工場の排気や排水の拡散のしかた、列車や自動車などの空気力学的特性には、すべて流体力学シミュレーションが利用される。希薄な気体の中を飛行するスペースシャトルのような宇宙機や、小さい構造物であるハードディスクのヘッドまわりには希薄流体シミュレーションが用いられる。ガソリンエンジンの内部の燃焼効率を上げるためには、反応入りの流体シミュレーションが必要である。また、電源トランスや磁石の設計では磁場のシミュレーションが行われる。携帯電話の電磁波が医療器具や航空機のシステムに干渉することが問題となっているように、情報化社会が進めば、ますます電磁波の影響を正確に予測することは重要に成ってくるであろう。また、日常的な天気予報や災害予測、最近とかく話題にされている環境問題など、物理シミュレーションの応用される分野はますます広がっている。

このような物理シミュレーションの多くは、メッシュを用いて行われている。しかし、上で見たように、計算機ハードウエアが 1000 倍高速化されても計算格子を増やすことによる離散化誤差の減少の効果は格子数でたかだか 3 倍程度しか期待できない。二次精度差分スキームを使って空間離散化の二乗で効果をあげることができたとしても離散化誤差の減少は高々10 倍程度しか期待できず、その効果は極めて限定的である。三次・四次精度のスキームを使ったとしても、計算の安定性のために局所的に精度を落とす必要があることが知られているので、複雑な構造物の周りでのシミュレーションを小さい離散化誤差で行うことに対する効果は限定的である。また、プログラマーの興味はより現実性を高めるため、新たな物理過程を導入することなどに注がれることが予想され、計算機能力の増大は新たな計算プロセスにもあてれることになる。

シミュレーションの分野で近年のハードウエアの進歩を十分に生かすためには、メッシュの張りなおしなどの操作を行うシミュレーションサポートツールは是非必要である。しかし、AMR ライブラリの作成にはシミュレーションの対象となる現象の知識、計算法の知識以外に高度なソフトウエア工学上の知識が必要となるため、従来シミュレーションを行っている研究者や技術者が手を出し難い分野である。また、AMR 法は、偏微分方程式で書ける問題に適用可能な、一般的な手法で

あるから、上であげたような物理法則の従う系の工学的応用のみならず、社会科学的な問題も含め、応用範囲は広いと予想される。

計算機の質的な変化という点に着目してみよう。従来の等間隔格子シミュレーションでは、データ構造も単純で計算手続きも単純なループだけで書くことができるので、ベクトルプロセッサーを搭載したスーパーコンピューターの独壇場であった。しかし、AMR 法は、空間にばらまかれた多数のメッシュ同士の関係を記述した、複雑な構造をもつデータの操作を行うために、 object 指向技術などの現代的手法を用いた方が、遙かに開発効率がよい。シミュレーションにメッシュ操作を組み合わせることになると、ベクトル計算機での効率向上は限定的なものになるが、一方で、目的の解像度を得ようとすれば、必要とされる演算量はスーパーコンピューターを必要とする程度のものである。近年は、ベクトル並列や高速 RISC チップの超並列などの並列機があるが、AMR 法は従来のベクトル機よりはむしろ並列機の上で実行することに適している。

現在のパーソナルコンピューターの高速化は目覚しいものがあり、個人ユースの廉価パソコンが 10 年前のスーパーコンピューターにも匹敵する演算性能を持っている。ただ、CPU は高速になったものの、パーソナルユースの計算機でのメモリーバス速度は、Pentium4 が登場したごく最近まで著しい高速化が行われていなかった。その点が、シミュレーションをパーソナルコンピューターで行うことに対する障害となっていた。現在オンチップキャッシュの搭載量も数 MB 程度にまで向上し、この容量は小さなメッシュ計算を行うには十分な容量となった。キャッシュまでの転送速度は、CPU の高速化に伴って非常に高速になっている。AMR 法は、格子数の比較的少ないメッシュ計算をたくさん同時に行う技術である。そこで、一時間ステップの発展に要する計算量が、メモリーバス速度に比べて十分多ければ、メモリー転送速度が低速であることはあまり問題にはならない。つまり、CPU の高速なパーソナルコンピューターは十分にスーパーコンピューターに対抗できる可能性がある。また、メモリーも我々が使用していた並列計算機での上限値は 8GB であり、パーソナルコンピューターでも 2GB を越えるメモリー領域をサポートし始めている。1PE あたりのメモリーバンド幅は、一部のマシンで数十 GB/s に達するもの存在するが、スーパーコンピューターであっても、1GB/s 程度のマシンは少なくない。一方、PlayStation2 や Pentium 4 に使われている RDRAM では、3.2GB/s を実現しており、数値的にはすでに高価な並列マシンを追い越している。近々発売予定のゲーム機 X-Box は、6.4GB/s というメモリー帯域を予告している。次に、並列化について見てみよう。パーソナルコンピューターやワークステーションの並列 CPU 仕様のものは、まだ高価である。しかし、たとえば 10Gbps の Ethernet の仕様が見え始めているので、ネットワーク結合を行った 1CPU パーソナルコンピューターでの並列計算でも、十分なパフォーマンスを出す可能性が見え始めている。このような構成であれば、価格的にも研究グループのレベルで十分現実性のある領域に入ってきているのではないか。今後、AMR 法のような計算方法に適した環境が到来しそうだと期待している。

2　Adaptive Mesh Refinement 法の歴史

2.1　Adaptive Mesh Refinement 法の分類

AMR は、格子生成法の一つと見ることができる。格子生成法といえば、従来はたとえば航空機の翼などの曲線の周りの流れを解析する場合、図 1 のようにその形状に沿った座標を形成するための技術であった。そして、このような格子を生成することが、数値計算法における一つの分野を形

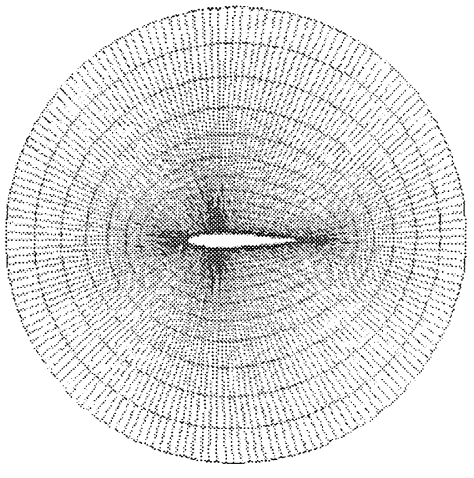

図 1:　航空機の翼の周りの流れを解析するための、従来の格子生成の例。
http://flow.dse.ibaraki.ac.jp/ plabo/AD1996/uzumaki/FIG2.GIF より引用。

成し、生成された格子が研究者間、あるいは開発者の間に有償で流通している場合も少なくない。ところが、3 次元計算を行う場合、また複雑な形状の場合、このような手段での格子生成は難しい。また、形状が移動する場合、移動に伴って格子を生成しなおさなければならず、計算効率が非常に悪くなる。

そこで、自動的に格子を生成してゆく技術が期待される。そういう技術として、Adaptive にメッシュを張りなおす方法が提案された。AMR 法と呼ばれる手法には大まかに見て 3 つの方法が考え出されている。一つは有限要素法を基礎としている手法、もう一つは構造格子 (直交座標系の単純な離散化と対応する格子、例えとしては、網戸の網目のようなもの) を用いた手法、そしてその中間的な方法としてセルツリーを用いる方法の、あわせて 3 つである。

有限要素法とは、空間を有限の三角形あるいは四角形に区切って、その上に微分方程式を組み入れてゆく方法である。たとえば図 2 のように空間を分割する。この分割に対して、解の精度を評価する方法があれば、精度不足の部分では現在の分割をもとにさらに細かい分割を行い、精度が十分のところはすでに分割した要素をひとまとめにして要素数を節約する手続きを行うことで、Adaptive Mesh といえるような方法になる。すでに有限要素法のコードを手にしているのであれば、有限要素法を用いた Adaptive Mesh 法は比較的簡単に実装可能だが、有限要素選択の自由度

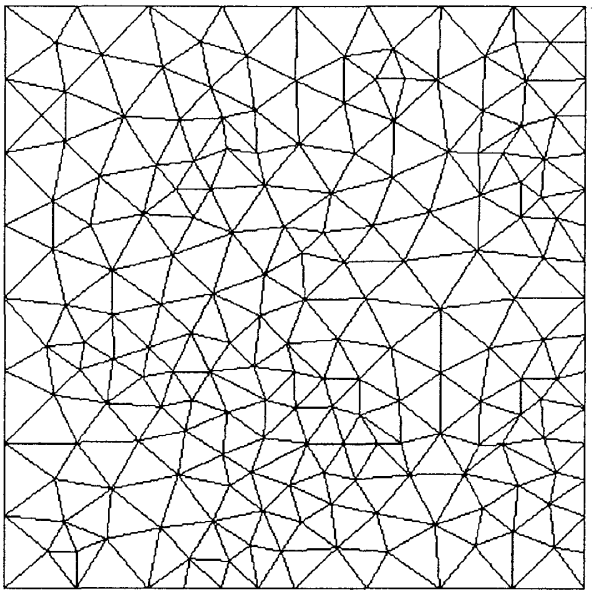

図 2: 有限要素の構成例。図は http://www.nag.com/simulation/Fastflo/ Documents/TutorialGuide/html/c05f01.gif より引用。

が高いため、(2) における、計算スキームの導出に、著しく強い制限を与えてしまい、スキームを選ぶ自由度が非常に乏しく、一般には陽解的なスキームしか用いることが出来ない。このため、精度向上や安定性の確保に、一般的には大きな制約を与えてしまう。

そこで、要素の形は四角形に限定し、セルのツリーを構成するような方法も提案されている。この方法は、de Zeeuw & Powell 1993[10] により提案され、 Melton & Berger & Aftosmis 1995[23], Khokhlov 1998[19] により実装され、日本の天文学研究者では、千葉大学の山下和之、国立天文台矢作、吉井譲のグループが採用している。Tree Base の AMR、Adaptively Refinement

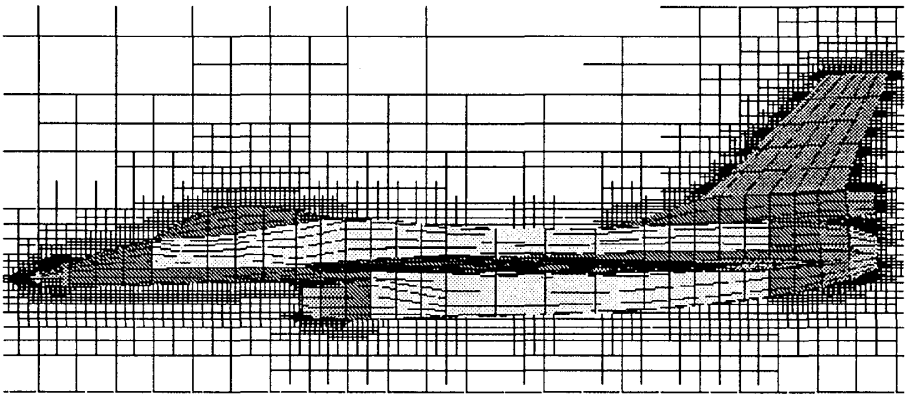

図 3: http://cs.nyu.edu/cs/faculty/berger/geom.html より引用。航空機の機体設計のために、セルツリーを構築した様子。

Mesh(ARM)、Unstructured AMR などと呼ばれている。しかし、この方法もデータ構造がセルのツリーであり、隣接格子がメモリー上の隣接点に存在しない場合も多く、データの配置を適切に

コントロールしなければ計算速度の向上は期待できない。また、このスキームの上で直接 Poisson 方程式を解く方法は、緩和法を用いれば可能だと思うが、日本のグループでは粒子的方法を併用することで Poisson 方程式を解いている。解くべき基礎方程式がさらに複雑なものになった場合に、容易に実装できるとは期待できない。格子間隔が異なる格子が隣接しており、精度が得られるかどうかについて評価した例を、筆者は知らない。図 3 に示す、航空機の設計に用いられている格子の例が、Web 上でアクセス可能なので、おおよそのイメージを持っていただけるのではないかと思う。

先ほどの物理シミュレーションの手順 (3) の観点からは、一般的な構造格子を用いた計算スキームの方が、望ましい。等間隔メッシュでのシミュレーションは、様々な分野で実績があり、またそのようなライブラリはすでに市販品や研究室などでも蓄積がある。この方法でのメッシュ生成は長方形の整合格子を基本とするため、従来の計算法上の様々な知識をそのまま適用できる利点がある。差分スキームの安定性や精度に関する議論も、このような格子を適用する場合には確立している。そこで、その点を考慮し、我々は、長方形のある程度まとまった格子数を持つ構造格子を Adaptive に張り直してゆく手法を用いている。図 4 に概念図を示す。この方法は、Berger

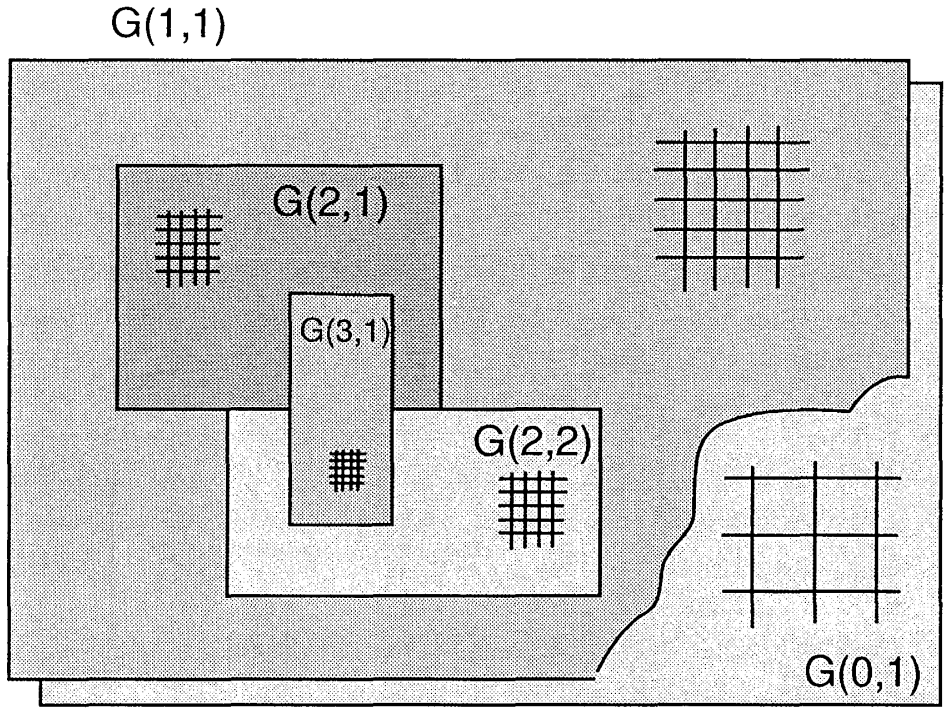

図 4: メッシュベースの Adaptive Mesh Refinement 法の概念図。各格子は $G(\ell, i)$ と記され、level ℓ と同じレベルの格子の index i の二つの index で区別される。G(0,1) と G(1,1) は、計算領域全体を覆うメッシュで、計算初期から最後まで変更されることなく存在する。隣接する二メッシュの比較により、必要な部分に通常のメッシュを生成する。

& Collera 1989[3], Berger & LeVeque 1989[4], Quirk 1991[26], Klein, McKee らのグループ [20], [31], M. Norman のグループ [24] により実装されている。Mesh Base の AMR、単に AMR、Structured AMR 等と呼ばれる。

しかし、個別のライブラリを AMR 法の枠組みに載せることは容易ではない。我々は、格子の生成消滅、物理量の新たに生成された格子への複写、同じ空間領域に重ねて張られた格子上での物理量の整合性の確保、格子の境界での境界条件の適用などこの方法特有の操作を、極力具体的なシミュレーションコードの内容と独立性を保った形でライブラリ化し、シミュレーションの汎用的な道具としての AMR ライブラリを作成しようと考えている。

なお、ここで整合格子という意味は、デカルト座標系での等間隔なメッシュに限定するものではない。物理座標としては不等間隔であっても、計算座標をデカルト座標系に map することができるものであれば、我々の AMR 法は適用可能である。例えば、図 1 のような座標系の上で、AMR 法を適用することができる。この場合、計算の中に座標系がデカルト的な意味で直交等間隔座標ではないことに起因する見かけの計量が入ってくることになる。そのような座標系を用いることによって、たとえば時間ステップの制限が計算空間上の一部分の振る舞いだけで決まってしまうということが起こりえる。そのことについて、ユーザーが熟知した上で使用するのであれば、適当な計算空間の上で AMR 法を適用することが可能である。

本稿では、以下我々の採用した方法に限定して述べることにする。

2.2 Mesh Base AMR の概要

上で示したように、我々が採用した Mesh Base の AMR 法は、複数の長方形格子の同時時間発展を行う方法である。それぞれの格子は、異なる解像度 Δx を持つ。解像度は、非負の整数 ℓ によって

$$\Delta x_\ell = \Delta x_0 u^{-\ell} \tag{1}$$

と書かれる。ここで、u は一つの計算の間には定数で、2 以上の整数である。この u を、upper level factor と呼ぶことにする。研究者によっては refinement ratio という呼び方を採用している場合もある。この式 (1) の ℓ は各メッシュの離散化度を表すもので、レベルと呼ばれる。レベルは、原理的に無限に増えてよいのだが、計算リソースの制限から無限に増えると計算が困難である。そこで、ℓ の最大値 $\ell_{\max}$ というパラメーターを用意しておくことにする。同じレベルの格子は一つとは限らない。そこで、それぞれの格子は、レベル ℓ と、同じレベルの格子を区別するための index i により

$$G_{\ell,i}$$

と書くことにする。また、同じレベルの格子の全体を、

$$G_\ell = \cup_k G_{\ell,k}$$

と書くことにする。

計算を開始するときには、格子は $G_{0,1}, G_{1,1}$ の二つしかない。これらの格子は、計算の全領域を D として、その全体をカバーする。

$$G_1 = G_{1,1} = G_0 = G_{0,1} = D$$

時間発展の途中では、たくさんの格子が存在する。実際、我々の計算でも 200 以上の格子が同時に時間発展している。隣接 2 層 (ℓ と $\ell+1$) で時刻が同期する毎に、結果を比較して精度評価を

行う。この結果、精度が不足しているなら $G_{\ell+2,i}$ を生成、精度が十分なら $G_{\ell+1,i}$ を消去してメモリーを節約する。この格子生成消滅の操作が終了した後、$G_{\ell+1,i}$ の結果を $G_{\ell,i}$ に上書きに上書きし、次の時間ステップの計算をはじめる。まとめると、図5のようになる。

点 $(x,y) \in D$ は、複数のメッシュに含まれることがある。この時、最終的な解は、この点を含む最も細かい格子からとられる。すなわち、最も細かいメッシュの解は、再計算されることがなければ求める誤差範囲で正しい解であるが、粗いメッシュでの解は求める誤差範囲を超えている場合があるので、採用されることの内容に注意を払うことが必要である。

2.3　アルゴリズム

この章では、図5に示す、従来用いられていたアルゴリズムに沿って、AMR 法のアルゴリズムを紹介する。我々が改良した点については、第4章に詳しく述べることにし、この章では問題点を指摘するにとどめる。

```
Procedure time_develop(level)
    do singlestep(level)
    better_boundary(level)

    if(next_level exists) then
        default_boundary(next_level)
        do r times
            time_develop(next_level)
        update(level)
    check_criterion(level)
    reconfigure_meshes(level)
```

図5: 従来のアルゴリズム

2.3.1　モデル方程式

従来の AMR 法を理解するために、以下のモデル方程式を考える。

$$\frac{\partial U}{\partial t} + \frac{\partial F(U)}{\partial x} + \frac{\partial G(U)}{\partial y} = 0 \tag{2}$$

流体力学の方程式は、このタイプである。ただし、U は、一般には多変数になる。ここでは、方程式は必ず流束関数 F および G でかかれることを仮定している。そうでない、一般的な問題の場合については、別途考察する。

2.3.2 境界条件と初期条件

計算領域 D の外の情報はユーザーが与える必要がある。メッシュ$G_{0,1}, G_{1,1}$ および、D の境界 ∂D に接したレベル 2 以上のメッシュに境界条件を与える手段は、一般のシミュレーション同様ユーザーが提供する必要がある。

計算領域の境界 ∂D に接していない、レベル $\ell > 1$ のメッシュ$G_{\ell,i}$ の境界 $\partial G_{\ell,i}$ の情報は、レベル ℓ の物理量自身から、あるいは必要ならば粗いグリッドの情報から線形に内挿して得る。そのためのもっとも単純な手続きは、ライブラリーが提供している。

計算精度の関係で、複雑な内挿式を用いる必要がある場合は、この手続きをユーザーが書き換えることができる。内挿方法は多種多様なので、ライブラリー設計としては、ユーザーが書き換える手段を提供する以上のことはできない。

各時間ステップの発展における初期条件は、メッシュの形が変わらない場合は、前の時間発展の結果を用いればよい。ただし、より細かいメッシュがある場合は、時間発展を開始する前に、細かいメッシュでの時間発展の結果を、適当な粗視化を行って上書きしておかなければならない。一般的な誤差評価方法は、一時間発展での誤差の程度を見積もるものである。最終結果としてより細かいメッシュの結果を保持しているとしても、誤差評価を適切に行うためには、毎時間発展ごとに粗いメッシュと細かいメッシュの間で結果の同期をとっておく必要がある。

新たに細かいメッシュが生成された場合は、適当な内挿手続きを用いて、現在ある粗いメッシュの結果を細かいメッシュに書き写す必要がある。この内挿手続きは、境界条件の場合と同様、ユーザーが提供すべきである。

レベル ℓ の長方形格子の境界条件と初期条件を決める手続きをまとめると、

(i) 境界セルを包含するレベル $\ell - 1$ の格子上の値から空間内挿して、レベル ℓ 相当に離散化された値を得る。
(ii) 必要ならば、適切な時間内挿を行い境界の解を得る
(iii) レベル ℓ に値を供給するさらに細かい格子があれば、(ii) から得られた値を上書きする。

ということになる。

2.3.3 時間積分

時間積分を行う場合、発展時間の決定と離散化を行わなければならない。

従来の多くの計算では、時間離散化と空間離散化の間に次の関係を仮定している。

$$\frac{\Delta t_\ell}{\Delta x_\ell} = \frac{\Delta t_{\ell-1}}{\Delta x_{\ell-1}} = \cdots = \frac{\Delta t_0}{\Delta x_0}$$

この点は、我々の改良点の一つであり、後述する。$\frac{\Delta t_0}{\Delta x_0}$ は (2) 式のようなタイプの方程式では、いわゆる CFL 条件で決まる。一般に、差分スキームを決めれば、Δt_ℓ の上限は決められる。

より細かい領域がない場合、式 (2) は

$$U_{i,j}^{n+l} = U_{i,j}^{n} - \frac{\Delta t}{\Delta x}\left(F_{i+1/2,j} - F_{i-1/2,j}\right) - \frac{\Delta t}{\Delta y}\left(G_{i,j+1/2} - G_{i,j-1/2}\right) \tag{3}$$

のように差分化される。差分の精度と安定性は、$F_{i\pm1/2,j}$ および $G_{i,j\pm1/2}$ の評価方法で決まる。これが、いわゆる差分スキームである。

複数の格子があるとき、境界値を決める以外は独立に時間発展する。ある格子を $t+\Delta t$ まで発展させるまでに、より細かい格子は時刻 t まで積分されていなければならない。はじめは式 (3) が適用される。以下の場合は修正を受ける。

(i) セルがより細かい格子に覆われている。
(ii) セルが細かい格子の境界に接しているが、それ自身は覆われていない。

(i) の場合は、レベル $\ell-1$ の格子での値は全てのレベル $\ell-1$ での積分が全て終わってからレベル l の格子の値の保存平均で置き換える。

$$U_{i,j}^{\mathrm{coarse}} \leftarrow \frac{1}{r^2}\sum_{p=0}^{r-1}\sum_{q=0}^{r-1} U_{k+p,m+q}^{\mathrm{fine}}$$

これは、粗い格子の流束を細かい格子の流束の各時間ステップの和として与えて、粗い格子で通常の時間発展を解くことと同等だが、この方法の方がメモリーの節約になる。

(ii) の場合、(3) を以下のように修正する。

$$U_{i,j}(t+\Delta t_{\mathrm{coarse}}) = U_{i,j}(t) - \frac{\Delta t_{\mathrm{coarse}}}{\Delta x}\left[F_{i+1/2,j}(t) - \frac{1}{r^2}\sum_{q=0}^{r-1}\sum_{p=0}^{r-1} F_{k+1/2,m+p}(t+q\Delta t_{\mathrm{fine}})\right]$$
$$-\frac{\Delta t_{\mathrm{coarse}}}{\Delta y}\left[G_{i,j+1/2}(t) - G_{i,j-1/2}(t)\right]$$

修正の実行の仕方は以下のようである。ある格子およびより細かい格子で (3) の形の積分が行われ、細かい格子での流束が全て知られているとする。粗い格子での流束が (3) にて計算された後、δF は次式で初期化される。

$$\delta F_{i+1/2,j} := -F_{l+1/2,j}^{\mathrm{coarse}}$$

細かい格子で各時間ステップが終了した後、δF は以下の修正を受ける。

$$\delta F_{i+1/2,j} := \delta F_{i+1/2,j} + \frac{1}{r^2}\sum_{p=0}^{r-1} F_{k+1/2,m+p}^{\mathrm{fine}}$$

u 段の細かい格子の時間ステップが終了したら、$\delta F_{i+1/2,j}$ を用いて粗い格子の解を修正する。

$$U_{i+1,j}^{\mathrm{coarse}} := U_{i+1,j}^{\mathrm{coarse}} + \frac{\Delta t_{\mathrm{coarse}}}{\Delta x_{\mathrm{coarse}}}\delta F_{i+1/2,j}$$

$i+2,j$ セルも refine されていたら、以下の修正も必要となる。

$$U_{i+1,j}^{\mathrm{coarse}} := U_{i+1,j}^{\mathrm{coarse}} - \frac{\Delta t_{\mathrm{coarse}}}{\Delta x_{\mathrm{coarse}}}\delta F_{i+3/2,j}$$

なお、境界流束 δF は細かい格子に随伴するベクトルとして保存される。

2.3.4 階層グリッドの生成と最適化

一定時間毎に誤差評価プログラムを走らせる。初期の実装では、最も粗い計算格子で 10 回程度の時間発展に対して一度程度、誤差評価と格子の再生成が行われていた。我々は、毎回誤差評価と格子生成を行っている。我々の主な改良点である、rewind と呼ばれる手法の実装のためには、毎回の誤差評価は必須である。

この手続きではさらに細かい格子を必要とする格子にフラグをたてる手続きと、重なりあう格子を最小に、不要に refine する格子数を最小に、パッチの数を最小にするなど、計算速度を最適化するために必要な手続きに分けられる。

格子生成に関しては、以下の基準を採用する。

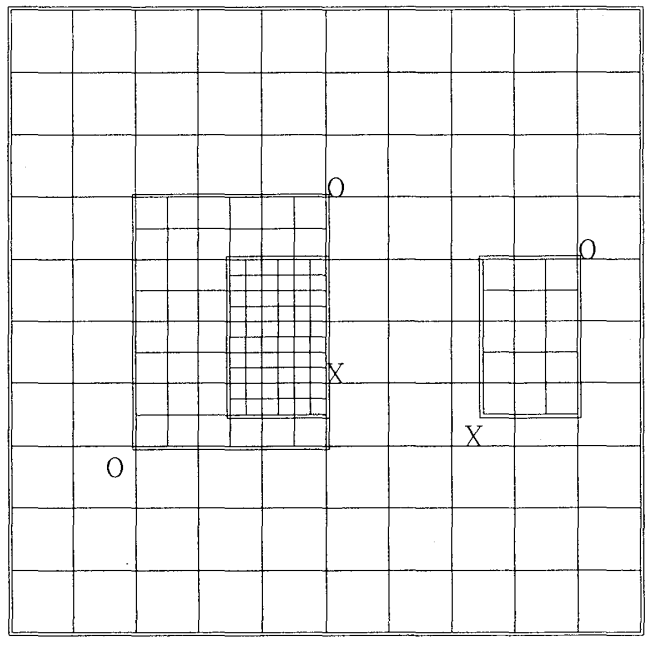

図 6: 格子生成の概念図。O で示したメッシュは構成されて良い場合、X で示した部分は生成のルールに反する場合を示している。左側のメッシュの右辺は、同じ線に沿ってレベルの二つ違う格子が隣接している。これは、本文中ルール (ii) に反する。右側のメッシュの左下角は、細かいメッシュの角が粗い格子の境界に沿っていないため、本文中ルール (i) に反する。

(i) 細かい格子の境界は次に粗い格子の角である。図 6 の右側のパッチの左下のような格子は作られない。

(ii) レベル ℓ とレベル $\ell-2$ の間に、東西南北方向少なくとも一つづつのレベル $\ell-1$ に属するセルがある。但し、そのセルが計算領域の物理的な境界である場合はこの限りではない。図 6 の左側のパッチの一番細かいメッシュは作られない。右側に少なくとも一つの中間の粒度のメッシュが入る。以後、この性質を proper nesting と呼ぶ。

この基準は、(i) はメッシュ境界での整合性を確保しやすくするためのものであり、流束等の保存量を扱う場合には特に有効である。(ii) は離散化度が急激に変化することによる精度の低下を防ぐための基準である。

具体的なメッシュ生成の手続きは、以下のようになる。

1. バッファー領域の生成 全ての格子にフラグのたっていないバッファー領域を付加する。次の regridding までに、誤差の大きい領域や不連続が、細かい格子から粗い格子へ抜けでてしまうのを防ぐ。双極系では情報の伝達は有限速度なので、バッファーゾーンを確保する方法で、情報が抜けてしまうことを防ぐことができる。2 ゾーンが典型的である。バッファー領域は高誤差のフラグをつけた領域の十分近くの点にフラグをつけることにより生成することもできるし [3]、メッシュを生成した後に領域を拡大することでも可能である。領域を後から拡張するのではなく先にフラグ領域を拡大してしておくことで、領域が重複する可能性を減少させることができる。
2. レベル $\ell+2$ の内点に対応する全てのレベル ℓ の点にフラグをたてる。 proper nesting を保証する。同じ理由で、G_ℓ の非物理的な境界はフラグをたてる点から除いておく。
3. 長方形の細かい格子を生成する。 全てのフラグのついた点を入力に、レベル $\ell+1$ の長方形の角を出力にする。詳細は以下に。
4. **proper nesting** を保証する。 もし proper nesting になっていなければ、長方形を細かくする。フラグをたてた点は proper nesting になっているはずだから、この手続きで絶対に proper nesting を保証できる。

最後の段階は、Berger & Colella[3] の実装では、bisection と mergins のステップから成り立っている。

はじめ、あるレベルのフラグのついたセルの周りに格子パッチが作られる。efficiency は、新しい格子のセルの総数とフラグのついたセルの数の比で定義される。この比が与えられた最低値 (例えば 60%) より小さいと、長方形の長い方が分割される。それぞれのパッチで最低値が保証されたところで、分割は終了する。これば bisection ステップである。

次に、merging を行う。merging はコスト関数の評価により行う。m 対 n の長方形では $(m+1)$ 対 $(n+1)$ の流束が計算され、演算量は流束の数にほぼ比例する。離散、収集の処理、境界条件、低レベルグリッドの値の置き換えなどのコストも計算する。ベクトル化効率などの要素も、コスト関数の決め方の中で考慮できる。

2.3.5　誤差評価

解 $U(x,y)$ が十分滑らかなら、空間間隔を h として、差分スキームをあらわす時間発展オペレーター Q が

$$\begin{aligned} U(x,t+k)-QU(x,t) &= k(c_1(x,y)k^q+c_2(x,t)h^q)+kO(k^{q+1}+h^{q+1}) \\ &\equiv \tau(x,t)+kO(k^{q+1}+h^{q+1}) \end{aligned}$$

で書けるとき、この差分スキームは q 次精度[*3]であるという。この時、

$$U(x,t+2k) - Q^2U(x,t) = 2\tau + kO(k^{q+1} + h^{q+1})$$

である。さらに、Q_{2h} をメッシュ間隔が $2h$ と $2k$ である Q と同じ差分スキームとすると、

$$U(x,t+2k) - Q_{2h}U(x,t) = 2^{q+1}\tau + O(h^{q+2})$$

従って、

$$\frac{Q^2U(x,t) - Q_{2h}U(x,t)}{2^{q+1}-2} = \tau + O(h^{q+2})$$

が成り立つ。AMR 法の実装では、この式を基準として誤差評価を小なうことが、一般的に行われている。

この方法の利点は、最終的に収束すべき値を知らなくても、また多くの点でのを調べて収束の様子を評価しなくても、隣接する二点の値をみれば、収束の度合いが分かることになる。[2] によると、この評価式を使う方法を Richardson 外挿と呼んでいるようである。その概念図を、図 7 に示す。

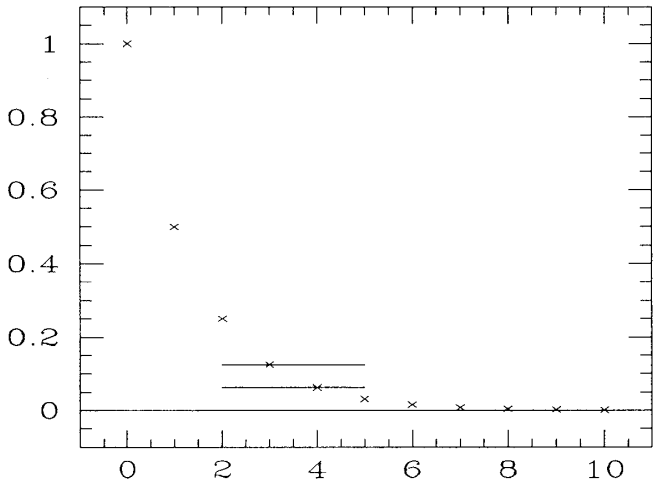

図 7: Richardson 外挿の概念図。真の値との差を縦軸にとっている。AMR においてはレベルに相当するパラメーターを横軸にとる。このパラメーターに対し、近似解の点列が得られたとする。隣接する二点の値を比べると、どの点においても、精度の良い方の解は隣接二点の差程度の精度で真の解と一致していることがわかる。

角計算ステップでは、あるレベルの格子点の値はな u 倍粗い粗視化格子に射影されている。もとの格子を u ステップ、粗視化格子を 1 ステップ時間発展する。粗視化格子のある格子点の誤差が許容誤差を越えたら、もとの格子の u^d 個の格子点ともフラグをたてる。ここで、d は空間の次元である。不連続のまわりでは、この方法は大きな誤差を与える。差分スキームの性質によっては、

[*3] この定義は通常 $q+1$ 次精度と呼ばれることが多いが、ここでは Berger & Colella[3] に従って、これを定義に採用する。

ショックのみは、$O(1)$ の不連続の近傍の誤差は無視して良い、すなわち refine する必要がない場合もある。slip 面や接触不連続を含む場合は無視できない。

強いショックを含む場合、$O(1)$ の誤差を含み、不連続をまたぐ特性線群に付随する $O(1)$ の波が発生する。単一格子では、この波はセルの端でキャンセルする。粗さの変わる点で、ショックは反射し、誤差が増大する。

2.4　格子分割集散法の検討

我々が採用した階層格子の最適化手法は、Berger& Colella[3] に紹介される、bisection と呼ばれる原始的な方法である。この方法では、不必要に小さな格子を作ってしまう可能性がある。そこで、Berger& Colella[3] では merging と呼ばれる方法を併用している。

より洗練された手法として、Friedel ら [12] は sawup と呼ばれる方法を提案している。

解の性質がよくわかっている場合、さらに効率の良い方法が考えられるかもしれない。この分野は、今後の進展が期待されるところである。

3　コード開発ポリシー

AMR 法は、非常に多数のメッシュの同時計算であり、物理量の整合性や時刻の同期、境界条件の設定など、複雑な手続きが多数存在する。科学技術計算用のソフトウエアであるという側面と同時に、巨大かつ複雑なデータベースであるという側面も持つ。また、精度の向上と実行効率を重視しなければならないという要求と、複雑なデータ構造を扱うためのわかりやすいプログラミングが望まれると言う要求もあり、この二つの要求は時に相反するものとなる。

このタイプのライブラリの開発は、「現象」、「計算法」、「ライブラリ設計」の３つの分野に関してそれぞれ深い専門的知識を持った上で、それぞれの要求の妥協点を模索してゆかなければならない。この三分野すべてに深い専門的知識をもつものは、容易に見つけられない。今までプロトタイプの開発において、開発者である我々二人はともに物理学の専門家であるとともに、山田は宇宙物理学における爆発現象の計算などで計算法の専門知識を持っており、また宮下は emacs を始めとする数々のソフトウエア開発に関わった経験からライブラリ設計に関する専門的知識を持っていた。つまり、二人ともが三つの分野のうち二つの分野において専門家であり、共通部分である物理という目的のもとに、もう一つの専門である数値計算法、あるいはライブラリー設計に対する相手の意見を十分尊重することで、この開発プロジェクトが動いてきた。これまでライブラリの中枢部分に関しては、二人が議論して合意することにより設計が進められてきて、このことが、計算方法上もライブラリ設計上も妥協できるバランスの良い設計を実現してきた。

さて、我々のライブラリ開発の目標は、高精度な計算をより高速に行うことである。精度を追求するには可視化が避けられないし、またより汎用性の高いものを書く場合ユーザーインターフェースやドキュメンテーションといった作業も必要となる。この部分は、現在とても手がまわっていない。我々二人がユーザーインターフェースや可視化のライブラリ開発に関わることは、ライブラリプログラム全体の開発効率から言って望ましいことではない。本稿の読者の中で、コードを利用す

るかわりに、ドキュメンテーションや可視化ライブラリーの作成などを担当してくれるものがいれば、筆者らとしては歓迎である。

我々のコード開発には、特に以下の点が重要視されている。

(A) 解くべき基礎偏微分方程式、また、そのための計算スキームから、ライブラリは完全に独立した設計にする。依存部が不可欠である時でも、適切な抽象化層 (abstraction layer) を用意し、上位層の依存を避けるようにする。

(B) ライブラリの使用者は、使用者の問題に関して選択された計算スキームに基づく実際のコード化の作業だけに集中できるようにする。つまり、計算スキーム及び、離散化の手法、変数の性質に基づく平均化の部分の実装だけが必要とされるように、充分なサービスを、一般性を失わない範囲で提供する。

(C) 性能、および、その拡張性 (scalability) を重視する。各 component は、出来るだけ効率の良い実装方法を追求する。また、並列化についても重視して設計しており、各 object の相互依存性を出来るだけ排除するように設計している。実際、現状で、POSIX thread library に基づく、並列化が実装されている。また、thread model に基づく並列化だけでなく、message passing model に基づく並列化も考慮中である。

(D) (C) に悪影響を及ぼさない範囲で、可搬性を重視する。中枢部のコードは、ANSI C++ および、ANSI C の標準を逸脱しないように書かれており、特定の machine architecture への依存は完全に排除されている。また、コンパイル作業も GNU autoconf を用いて可搬性を高めており、一般的な POSIX system には、容易に移植することが出来るようになっている。また、Windows 上でも Visual C++で動作するようになっている。

(A) の独立設計のため、流体力学、宇宙物理学、相転移問題、燃焼問題など、さまざまなスキームに同じプログラムが利用可能である。しかし、(A) の設計方針は、応用可能性を目的としたものではなく、応用可能性は副産物としてついてくるものである。このような、高度なプログラミングにおいては、分離可能なものは極力分離し、抽象的な形で実現しておかなければ、コードのバグを発見することも著しく困難になってしまうのである。

(C) の設計はバランス問題である。コードを洗練されたものにすることにより、実行効率が犠牲になるようなことは、極力避けている。そのことが、データ構造にある種の制限を加えてしまう場合もある。しかし、効率が必要とされ、かつ効率に非常に関わる部分では、それもやむを得ない。また、実際に計算を行った結果、並列化効率も非常に良いことが分かった。ただ、POSIX thread library に基づく並列化を行っているため、thread の同期に対するオーバーヘッドが大きく、共有メモリー型の Simmetric Multi Processing(SMP) 型の計算機以外での効率は、今のところ評価していない。

ベクトル化効率について、我々の Mesh base の実装法では Adaptive 操作とベクトル化は関係ない。ユーザーがライブラリーの下に実装する差分コードがベクトル化されていれば、その範囲でベクトル化は行える。ただ、実際には C++ と Fortran の linkage は問題ないはずだが、POSIX thread library による並列化と Fortran の相性が悪い場合がある。また、C++や C の処理系が構

造体を含むデータに対して効率の良いベクトル化コードを出力しない場合がある。この両方の事情が同時に起こった場合、ベクトル化と並列化の両立は困難である。すなわち、上の二つの事情が重なった場合には、C++ではベクトル効率が出ず、Fortran を併用すると並列化に支障があるため、ベクトル並列機では性能を出すことができない。世の中のスーパーコンピューターが、早くC++/C での自動ベクトル化をちゃんとサポートしていただきたいと、切に願っている。

(D) の設計手法のおかげで、現在 Linux、FreeBSD、IRIX、Digital Unix 上で動作している。スーパーコンピューターの利用目的のため、ほかの特殊な OS を使わなければならないとしても、標準的な POSIX 環境であれば、移植性はかなり高い。autoconf には頼っていないが、現在独立に Windows 系の OS の上で Visual C++が動作する環境であれば、そのための Makefile も同時に構築している。

4　精度の再評価と我々の改良点

4.1　時間ステップの柔軟化

先にも述べたように、多くのコードは $\Delta t_\ell = \Delta t_0 \frac{\Delta x_\ell}{\Delta x_0}$ を仮定している。また、$\Delta x_\ell / \Delta x_{\ell+1} = 2$ で固定のものまである。我々は、このパラメーターは問題依存で変更できるべきものであると間が手いるので、当然変更可能である。

時間ステップに対する仮定が妥当ではないことは、たとえば衝撃波管問題などを考えてみれば自明である。ステップ毎、レベル毎に独立に CFL 条件を課すべきである。

海外の研究者が公開したコードを利用すればよいという意見もある。しかし、自分で実装してみなければ、コードの性能や限界について意見が言えない。Adaptive Mesh に関して、精度が出るとか出ないとかさまざまな意見があるようだが、精度が出ないとすればどういう理由なのか、それは改良可能なのかを検討することができるためには、実装してみるか、公開コードを使用するにしても、少なくともすみずみまでコードを理解している必要がある。他人のコードをすみずみまで理解するのは、そのコードの出来がよほどよくなけば難しく、自分でコードを書いたほうが早いことは多々ある。また、将来ほかの研究者が意見を言えるように、われわれのコードは現代的な意味でのコード設計手法にしたがって作られていることを付記しておく。コードを公開するというのは、そういうレベルで行われるのでなければ、自己満足以上の何者にもならない。

時間ステップの柔軟性は、実装上の困難の度合いで言えば、マイナーな問題である。時間発展のアルゴリズムが、反復回数のカウントによる単純なループではなく、条件判断を含む多少複雑なループ、あるいはリカーシブになる程度である。

4.2　時間発展スキームの改善

図 8 に、従来の時間発展法と我々の改良した時間発展方を示す。図中の数字は、時間発展手続きが行われる順序である。従来法では、1, 2, 3 の時間発展を行った後に誤差評価をし、精度が不足していれば、1 および 3 の計算が終わった時刻で細かいメッシュを生成し、ここから計算をはじめ

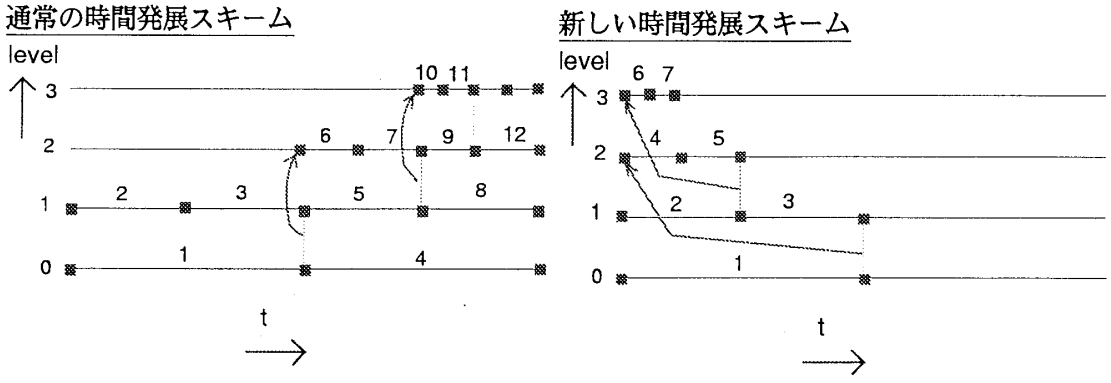

図 8: rewind の概念図：左図で 0 と 1 のレベル間で、誤差が閾値を越えるとレベル 2 を遡って積み上げる。中央の図の様に新たに各レベルを時間発展させる。更に、1 と 2 のレベル間で誤差が閾値を越えると、更に遡ってレベル 3 を積み上げる。このようにして、誤差が閾値を越えないように時間発展を行う。各図中の $1, 2, 3, \ldots$ は、実際のステップ発展の順序を表している。上層のレベル程時間刻が短い事に注意。

る。この場合、2, 3 の計算を行った離散化度では精度が不十分である。そして、6 の計算を開始するために用意される初期条件は、3 の終了時のデータから得られる。つまり、6 の計算の初期条件にはすでに大きな誤差が含まれている。したがって、この部分の誤差はそのあとの計算で改善される可能性は、偶然正しい解に向かう以外にはあり得ない。そこで我々は、誤差評価の結果メッシュの再構築が必要であると判断した場合には、それぞれの計算が誤差評価を行った時刻の一つ前に同期した時点にさかのぼって、計算を再度行うような改良を行うことにした。これを、rewind と名づける。

Berger & Oliger の論文示されているアルゴリズムは不明瞭で、どちらをともとれるのだが、Berger & Oliger の論文にはデータ構造が示されており、このデータ構造で実装可能なアルゴリズムは、我々が図中で従来法としてしめしたものだけがである。Berger & Oliger の論文だけ引用して我々の改善が新しくないと主張するのは、誤りである。実際、多くの研究者がこのあとの計算で具体的なアルゴリズムを示しており (図 5 等)、すべて我々が従来法として示したものになっている。我々の改良スキームを、図 9 に示す。

我々は、この点が AMR 法の精度に大きくかかわっているのではないかと考え、精度チェックを行った。この方法の黎明期には、たしかに収束性のチェックがいくつか行われている。しかし、これらの精度チェックは $\ell \leq 3$ でしか行なわれていない。しかし、実際のシミュレーションでは 10 段 15 段といった非常に大きなレベルを用いた計算が行われているにも関わらず、その結果の正当性に関しては、適切な精度の検証が全く行われないまま議論が展開されている。我々は、従来法ではレベル数が 3 程度までしか期待される精度が得られず、レベルの数を増やす場合、我々の提案した新しい時間発展スキームを使わなければならないことを、具体的に衝撃波管問題と Burgers 方程式の数値実験により明らかにした。

```
Procedure time_develop(level)
  if(! stored(next_level)) then
    save(next_level)
  while(need)
    do singlestep(level)
    better_boundary(level)
    if next_level exist; then
      default_boundary(next_level)
      time_develop(next_level)
      save(next_level)
      if (level > 0); then
        need = check_criterion(level)
        if (need)
          restore(level)
          reconfigure_meshes(level)
          store(level)
        else
          reconfigure_meshes(level)
```

図 9: 我々が改良した、新しい時間発展スキーム。

4.3 新しいスキームの困難

改良時間発展スキームでは、時間を戻すために、時間発展の履歴を覚えている必要がある。我々は、このためにデータ構造の変更を行っている。メモリー使用量は、原理的にはもとのスキームの数倍にはね上がるが、実際には 2 倍程度に収まっている。この手法を、我々は rewind と名付ける。これは、従来のスキームでは誤差が蓄積され、上のレベルのメッシュを作成する必要があると判定される領域が広がるが、時間発展スキームを改良したことの副作用でメッシュが張られる領域が小さくなったことによるものと考えている。

4.4 テスト問題

精度評価のためのテスト問題として、我々は衝撃波管問題と Burgers 方程式を採用した。この他よく用いられるテスト問題として、スカラー移流型方程式も試したが、これに関する結果は他の結果と重なる部分があるので、ここでは割愛する。

問題選択の基準としては、常微分方程式に帰着する、あるいは厳密積分形に帰着する場合も含めて厳密解が存在すること、物理数値計算で問題となる性質を持つこと、単一メッシュの計算との比較可能な問題設定であることを重視した。以下に、それぞれの問題の基礎方程式を書く。

衝撃波管問題は、Sod 問題 [28]、あるいは Riemann 問題等とも呼ばれ、方程式と初期条件は以

下の形になる。

$$\frac{\partial}{\partial t}\begin{pmatrix}\rho\\ m\\ E\end{pmatrix}+\frac{\partial}{\partial x}\begin{pmatrix}\rho u\\ \rho u^2+P\\ u(E+P)\end{pmatrix}=0$$

$$m=\rho u,\ E=\rho e+\frac{1}{2}\rho u^2,\ \gamma=1.4,\ P=(\gamma-1)E$$

$$\begin{pmatrix}\rho\\ u\\ P\end{pmatrix}_l=\begin{pmatrix}1.0\\ 0\\ 1.0\end{pmatrix},\begin{pmatrix}\rho\\ u\\ P\end{pmatrix}_r=\begin{pmatrix}0.125\\ 0\\ 0.1\end{pmatrix},$$

ここで、基礎変数としては、保存量である密度と運動量、エネルギーを採用している。これは、衝撃波を含む場合に差分解が求めるべき物理的な解に収束するための条件として、保存変数を用いることが必要であることが証明されている [22] ためである。なお、衝撃波の両側の値としては、SOD の採用した値をそのまま用いている。この問題では、衝撃波不連続、接触不連続、膨張波など、Euler 方程式を数値的に解く上で技術的困難がある多くの要素を、同時に含んでいる。この問題を精度良く解くことができるかどうかは、手法の選択上重要な基準となる。歴史的にも、良いチェック問題として多くの研究者がこの問題を使ってきた。

Burgers 方程式 [6] は、以下の形の方程式である。

$$\frac{\partial}{\partial t}\boldsymbol{u}+(\boldsymbol{u}\cdot\nabla)\,\boldsymbol{u}=\mu\nabla^2\boldsymbol{u}$$

この方程式系は、任意次元で意味を持ち、かつ厳密解を持つ。従って、多次元の場合の最も簡単なチェック問題となる。また、粘性計数 μ が 0 でない場合でも厳密解が存在するので、スキームに内在する粘性の効果を評価するにも適している。初期条件としては、

$$\boldsymbol{u}(\boldsymbol{r},0)=\begin{cases}\boldsymbol{0} & (|\boldsymbol{r}|<r_1)\\ h_0\dfrac{\boldsymbol{r}}{|\boldsymbol{r}|} & (r_1\le|\boldsymbol{r}|\le r_2)\\ \boldsymbol{0} & (|\boldsymbol{r}|\ge r_1)\end{cases}\tag{4}$$

を用いた。

厳密解と評価された精度については、以下に二つの sub section を作って述べることにする。

4.5 流体の計算結果

SOD 問題 [28] については、厳密解は良く知られているのでここでは省略する。この問題について、我々は

$$u=2:u^\ell=8\sim512,u=4:u^\ell=16\sim1024$$

のパラメーターを用い、また計算のためのパラメーターとして CFL 数=0.9、および refinement ratio u と最大レベル ℓ_{max} は計算ごとに違う値を用い、その依存性を評価した。差分スキームは、Roe 法 [27] に二次精度 MUSCLE[36, 35, 34, 33, 32] の手法を併用した方法を用いた。時間発展手法は *rewind / no rewind* で示した。*rewind* が我々の実装、*no rewind* が従来の実装である。

誤差評価基準としては、次のようなノルムを定義するのが望ましいと思われる。

$$\int\left(\left(\frac{|\rho_\ell-\rho_{\ell-1}|}{\rho_{\max}}\right)^2+\left(\frac{|m_\ell-m_{\ell-1}|}{m_{\max}}\right)^2+\left(\frac{|E_\ell-E_{\ell-1}|}{E_{\max}}\right)^2\right)dx$$

厳密解がわかっているので、$\rho_{\max}, m_{\max}, E_{\max}$ は計算可能だが、どれもオーダー $O(1)$ の量なので、ここでは最大値ではなく 1 で規格化した以下の式を用いた。

$$\int \left(|\rho_\ell - \rho_{\ell-1}|^2 + |m_\ell - m_{\ell-1}|^2 + |E_\ell - E_{\ell-1}|^2 \right) dx$$

メッシュの分割は、誤差の多い格子点の数が全体の 0.6 より少なければ再分割を行うようにした。

結果を、我々は誤差とメモリーや計算時間といった計算リソースの消費量との比較で示すことにする。というのは、我々が欲しいのは具体的な計算パラメーターではなく、ある精度の計算を行うのにどういう方法が効率的であるかという基準であるからである。結果を図 10 に示す。

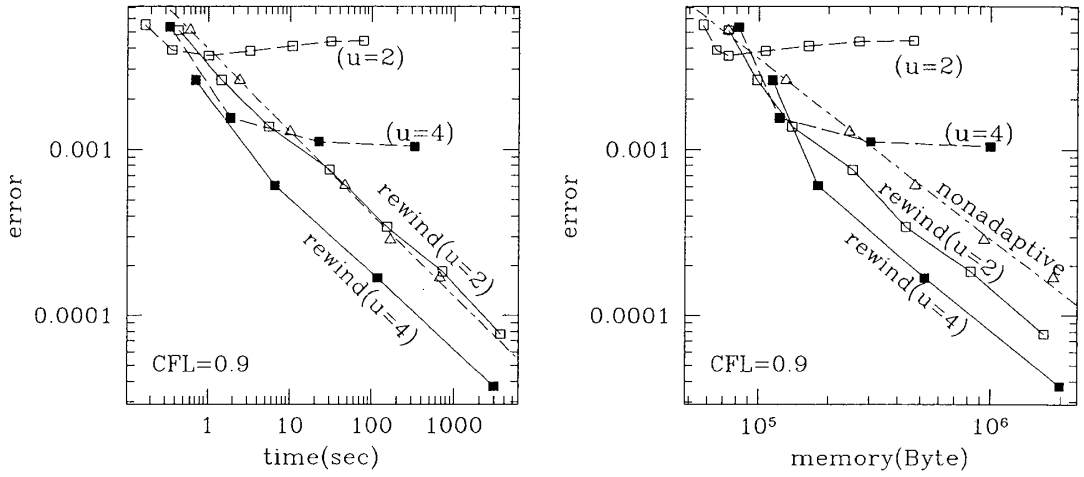

図 10: SOD 問題における、メモリーおよび計算時間と誤差の関係。左下側にいくほど効率がよい計算であることを示している。誤差は L^2 ノルムで評価した。白の三角形はは、AMR 法を用いない通常のメッシュ計算で、メッシュ数は 800, 1600, 3200, 6400, 12800, 25600 である。nonadaptive と書いた一点鎖線は、best fit の直線である。四角で書いたものは、AMR 計算によるものである。実線は、rewind を行った場合、破線は rewind を用いなかった場合である。u の値は、upper level factor である。白い四角形は $u = 2$ の場合、黒い四角形は $u = 4$ の場合にそれぞれ相当する。格子数は、level$\ell = 0$ のメッシュで 100、ℓ_{max} は $u = 2$ の場合が 3, 4, 5, 6, 7, 8, 9、$u = 4$ の場合が 2,3,4,5 である。

この結果を見やすくするために、式 (5) で示す error equivalent levelℓ_{eq} という概念を導入することにする。これは、単一メッシュの計算結果の誤差から、計算をある誤差以内で行うために必要な単一メッシュ計算のメッシュ数を割り出し、これをレベルという概念で焼き直したものである。図 10 の nonadaptive の直線上に書いてある三角の記号の並びは、メッシュ数を等比級数的に増やした場合、あるいは格子間隔 Δx を等比級数的に減らした場合のものである。これがこの図でほぼ等間隔に見えると言うことは、誤差はメッシュ数の適当なべきで書けることを意味する。SOD 問題の我々の計算例では、ほぼ

$$\varepsilon = 4.14 \times 10^{-2} \Delta x^{1.007234},$$

でフィットできる。この問題は不連続を伴う問題であると言うことである。不連続を伴う問題の場合、双曲型方程式の数値解法では安定性のため不連続面付近は一次精度で解く。このべきが 1 に近

い値をとっているのは、計算スキームのせいではなく、双曲型方程式である以上さけられない事情によるものである。

この式から、逆に誤差を決めると、その誤差で計算可能な単一メッシュ計算での格子間隔が計算できる。この格子間隔を AMR 計算上での最もレベルの高いメッシュでの格子間隔に対応させるレベルに対応付けすることを考えると、以下のような式になる。

$$\ell_{\mathrm{equiv}} = -\log_u \frac{\Delta x}{\Delta x_0} \sim -\log_u \frac{23.6 \times \varepsilon^{0.993}}{\Delta x_0}. \tag{5}$$

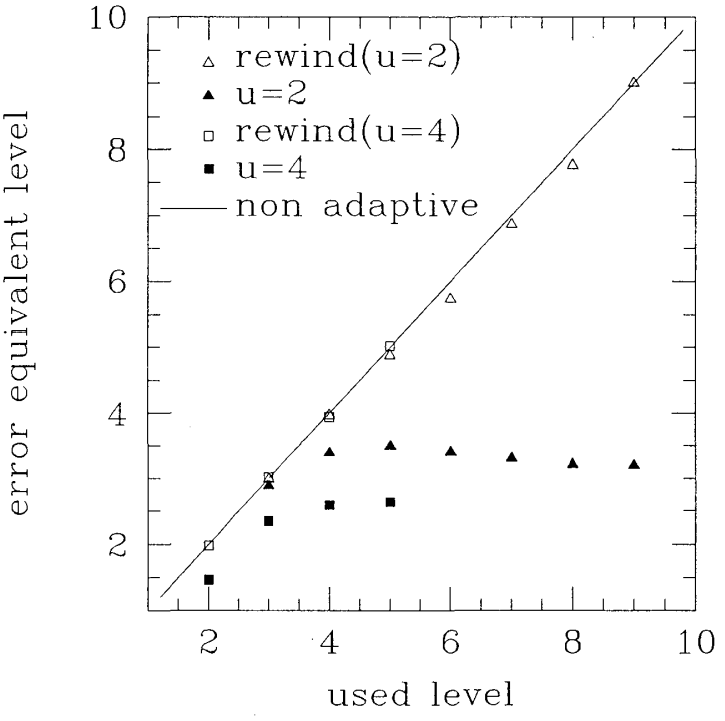

図 11: 横軸は、rewind を行った場合および行わなかった場合に、AMR 計算で設定した ℓ_{max} を示す。縦軸は、計算誤差を示す。軸の値は、本文中で示した式 (5) に従って、誤差から評価した値である。この図では、左上が効率と精度の良い計算、右下は効率と精度の悪い計算である。定義から、nonadaptive の計算では実線で書いた $x = y$ になる。

図 11 に、実際に使ったレベルの数と、上で評価した error equivalent level との比較を示してある。理想的な計算では、勾配一の直線に乗るはずだが、従来法ではレベルが 3 程度で誤差が収束し、それ以上計算結果は改善しないことを示している。一方我々の計算では、ほぼ理想的な計算に相当する直線に乗っている。

4.6 Burgers 方程式の計算結果

Burgers 方程式

$$\frac{\partial}{\partial t}\boldsymbol{u} + (\boldsymbol{u}\cdot\nabla)\,\boldsymbol{u} = \mu\nabla^2\boldsymbol{u}$$

を、渦がないと仮定し、適当なポテンシャル $\phi = \nabla\phi$ を設定して、Cole[7] Hopf[15] 変換

$$\boldsymbol{u} = \nabla\phi,\ v = \exp\left(-\frac{\phi}{2\nu}\right)$$

により v の方程式に変換してみる。すると、この v は、熱伝導方程式を満たすことがわかる。熱伝導方程式は、

$$\begin{aligned} v(0,\boldsymbol{y}) &= \exp\left(-\frac{1}{2\nu}\int^{\boldsymbol{y}} \boldsymbol{u}(0,\boldsymbol{y}')\cdot d\boldsymbol{y}'\right) \\ \boldsymbol{u}(t,\boldsymbol{x}) &= \int_{R^n} \frac{\boldsymbol{x}-\boldsymbol{y}}{t}\exp\left(-\frac{G}{2\nu}\right)d\boldsymbol{y} \\ &\quad \times\left(\int_{R^n}\exp\left(-\frac{G}{2\nu}\right)d\boldsymbol{y}\right)^{-1} \\ G(t,\boldsymbol{y},\boldsymbol{x}) &= \int_0^{\boldsymbol{y}} \boldsymbol{u}(0,\boldsymbol{y}')\cdot d\boldsymbol{y}' + \frac{|\boldsymbol{x}-\boldsymbol{y}|^2}{2t} \end{aligned}$$

の積分形の厳密解を持つことが知られている。

渦がないと仮定することに付いては、簡単な式変形により、

$$\begin{aligned} 0 &= \frac{\partial}{\partial t}\boldsymbol{u} + (\boldsymbol{u}\cdot\nabla)\,\boldsymbol{u} - \mu\nabla^2\boldsymbol{u} \\ &= \frac{\partial}{\partial t}\boldsymbol{u} + \frac{1}{2}\nabla|\boldsymbol{u}|^2 - \boldsymbol{u}\times(\nabla\times\boldsymbol{u}) - \mu\nabla^2\boldsymbol{u} \\ &= \frac{\partial}{\partial t}\boldsymbol{u} + \frac{1}{2}\nabla|\boldsymbol{u}|^2 - \mu\nabla^2\boldsymbol{u} \end{aligned}$$

が導かれる。ここで、

$$\boldsymbol{u} = \begin{pmatrix} u \\ v \end{pmatrix}$$

である。

$$\begin{aligned} \frac{\partial u}{\partial t} + \frac{\partial}{\partial x}\frac{u^2+v^2}{2} &= \mu\frac{\partial^2}{\partial x^2}u + \mu\frac{\partial^2}{\partial y^2}u \\ \frac{\partial v}{\partial t} + \frac{\partial}{\partial y}\frac{u^2+v^2}{2} &= \mu\frac{\partial^2}{\partial x^2}v + \mu\frac{\partial^2}{\partial y^2}v \end{aligned}$$

この方程式は、初期に渦無し条件を満たせば任意の時刻で渦なしであることが知られている。

差分化は、Cranck Nikolson スキーム [9] を用いた。

図 12 に、式 (4) で示される初期条件と $t = 0.7$ での厳密解 $\boldsymbol{u}$ の動径成分の動径分布、および計算終了時のメッシュ分布の様子を示した。解の動径方向にシャープなフロントを持ち、外側に広がってゆくタイプのものだが、前面のシャープなフロント付近に沿って、細かい格子が張られており、計算リソースを有効に使っていることがわかる。

SOD 問題の場合と同様、計算時間とメモリー効率の比較を行っている。no rewind 計算では、nonadaptive 計算よりむしろ効率が悪く、十分に計算精度が上がっていない。一方 rewind 計算では、ℓ_{max} を大きくしてゆくに従い、nonadaptive 計算より左下側に大きく離れた点に、データ点が分布しており、効率の良い計算ができていることがわかる。

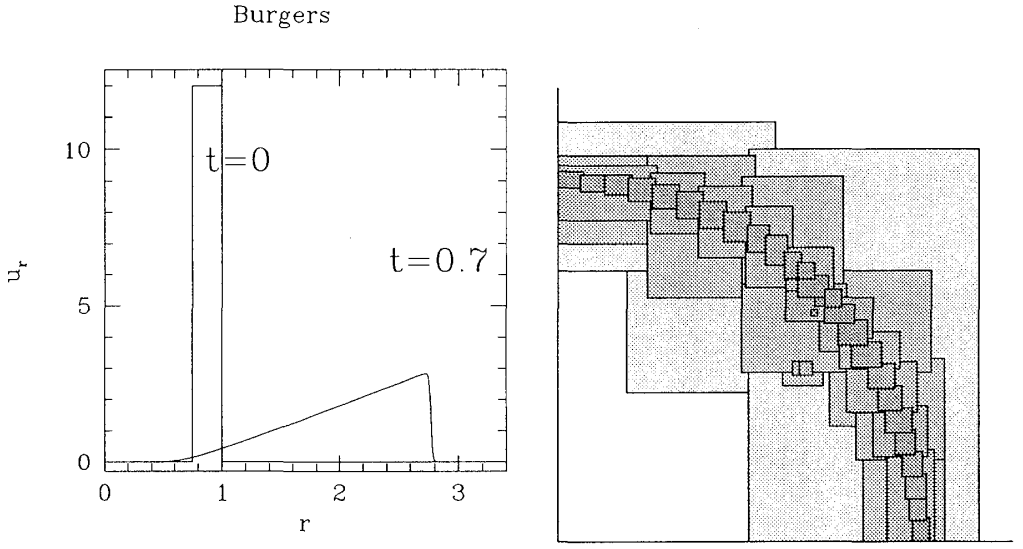

図 12: Burgers 方程式の初期条件と厳密解 (左) と、メッシュ構成例。厳密解の図で、t は、この方程式での時刻を表す。u_r は、$\boldsymbol{u}$ の動径成分を表す。粘性 μ の値は 0.01 にとった。メッシュ構成例は、それぞれの長方形が一つのメッシュの位置に対応する。長方形の灰色のトーンは、レベル ℓ が大きいほど濃くなっている。レベル 2-5 を示している。レベル 0 と 1 は、計算空間全体を覆っている。

rewind を用いない計算で、計算精度が出ていないように見えるのは、シャープなフロントでの計算精度の低下に原因がある。従来法の精度の悪さを極力強調しないように、我々は Crank-Nicholson を選択したが、スキームとしてもっと robust な TVD スキームを用いれば、この差はもっと大きくなる。メッシュ計算では、数値安定性のため、このような不連続面の前後で多少粘性が入ることは避けるれない。従って、シャープなフロントの高さは、TVD スキームでは若干低く押さえられることになる。このフロントは、厳密な意味で不連続面ではないが、不連続面と考えてフロントの移動速度を評価すれば、これは前後の物理量の飛びで決まる。フロントが低く押さえられると言うのは、フロントの移動速度を過小評価することにつながる。フロントの位置を正しく評価できないと、L^2 ノルムでの誤差評価は非常に大きな値になってしまう。

これは、誤差評価方法が悪いという考えもあると思う。しかし、フロントの影響の広がり方が重要な問題というのは、少なくない。たとえば環境問題などをシミュレーションする場合の有害物質の広がり具合は、成分の異なる層の境界の移動を問題にする。爆薬の効果や工場などの爆発事故の影響がどの程度あるかを調べる場合は、まさに爆発面の進行状況を正しく評価する必要がある。燃焼系に適用する場合は、補正がどちら側に倒れるかがわからないと言う点から、この事情は非常にシビアなものになる。というのは、爆轟波の移動を過小評価すれば、爆轟波の移動速度が小さくなる用にも思えるが、逆に爆轟波後面の圧力を過大評価して現実以上に加速するという解釈があり得るからである。従って、不連続面の位置決定の誤差を大きく拾うような誤差評価方法も、計算の適用を考える上で、決して恣意的なものではない。

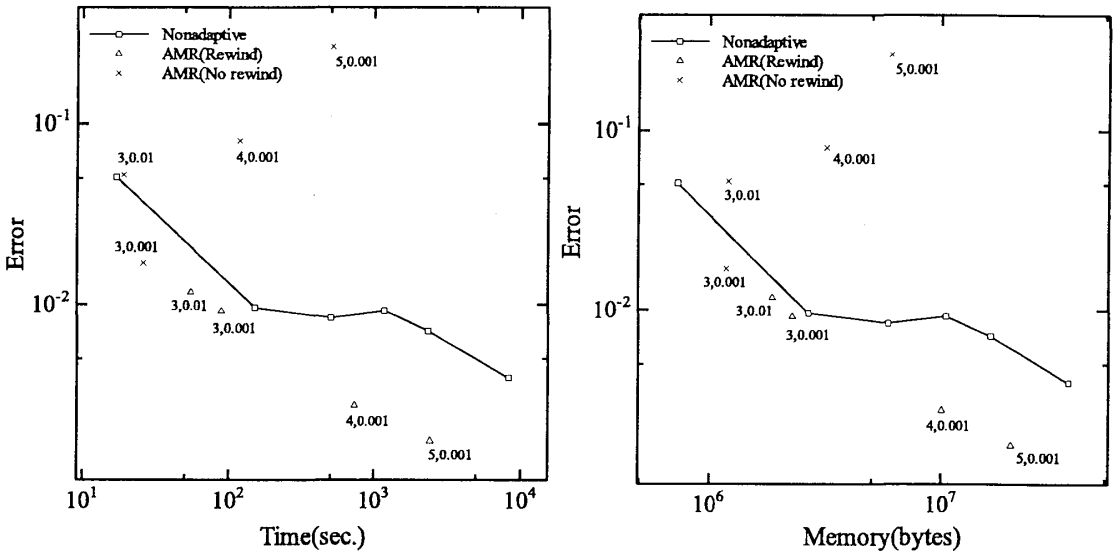

図 13: 計算時間と誤差 (左) および計算時間とメモリー (右) の関係を示す。格子数は、AMR 計算では $\ell = 0$ で 30×30、nonadaptive 計算では、120×120, 240×240, 360×360, 600×600, 900×900 をそれぞれ用いた。nonadaptive 計算の結果は線で結んである。AMR 計算の結果は、ℓ_{max} と誤差の最大値を示すパラメーターを、点の横に記した。

5　流体力学への応用

この章では、AMR 法の応用例として、２次元非粘性ブシネ近似方程式の有限時間発散の可能性を調べたので、この結果について紹介する。この手法を用いると 1 次元方向に最大 2^{21} 個に相当する実効メッシュを用いた計算が可能である。この結果、温度勾配と渦度の絶対値の最大値がそれぞれ $(t_0 - t)^{-2}$、$(t_0 - t)^{-1}$ で発散することを示唆する結果を得た。発達の初期には温度渦層が形成されるが、これは高々指数関数的増大過程である。その後、層の不安定化に由来すると思われる空間変動が見えるようになり、有限時間発散過程に移行すると考えられる。

5.1　はじめに

乱流の普遍性は、コルモゴロフの局所等方仮説に基づくカスケード理論によって認知されるようになった。この仮説では、エネルギー散逸率と粘性が独立であるとしている。非粘性極限においてもこの仮説が成立するためには、エンストロフィーが発散することになる。このことから、非粘性での解の発散問題が古くから注目されて来た。また、その後の間欠性研究においても、仮想的な速度場の自己相似的な特異性の存在が仮定され、非粘性極限であるオイラー方程式の相似性との関連で理解されている。以上のように、乱流の普遍的性質と非粘性系の解の有限時間発散との関連性から、その可能性を探る研究は少くない。しかしながら、計算機能力の限界と理論的な裏付けが得られないことから決定的な結論は出ていない。さらに、最近の直接シミュレーションを用いた乱流場の詳細な研究により、特異性があったとしても乱流特性への直接的な影響は小さいと考えられるよ

うになってきた。

2 次元ブシネ近似方程式の乱流と非粘性での特異性の可能性については、最近藤、松本、鈴木らのグループ [30, 29] で調べられている。ここでは、さらに AMR 法で、この様子を詳しくかつ直接的に調べることが、目的である。この章の研究は、山田、宮下、藤、松本の 4 人の共同研究である。2次元ブシネ方程式 (以下 2DFC と略す) は、以下の様に浮力項のみ圧縮性を考慮するブシネ近似を用い温度と速度場が結合している：

$$\frac{\partial T}{\partial t} + \boldsymbol{v}\cdot\nabla T = \kappa\triangle T, \tag{6}$$

$$\frac{\partial \boldsymbol{v}}{\partial t} + \boldsymbol{v}\cdot\nabla \boldsymbol{v} = -\nabla p - \alpha g T \boldsymbol{e}_y + \nu\triangle \boldsymbol{v}. \tag{7}$$

ここで、ν と κ はそれぞれ動粘性率と熱拡散係数で、α、g は体積膨張率と重力加速度の大きさを表し、$\boldsymbol{e}_y$ を y 軸方向の単位ベクトルとすると重力の方向は y 軸下向きとなる。一般性を失うこと無く αg を 1 とできる。また、既に平均密度 ρ_0 は、1 と置いている。

理解を助けるために、新たにベクトル量 $\boldsymbol{\chi} = (\frac{\partial T}{\partial y}, -\frac{\partial T}{\partial x})$ を導入する。便宜上、このベクトルを T 渦度と呼ぶ。T 渦度と渦度の従う方程式は、以下のようになる。

$$\frac{\partial \boldsymbol{\chi}}{\partial t} + \boldsymbol{v}\cdot\nabla\boldsymbol{\chi} = \boldsymbol{\chi}\cdot\nabla\boldsymbol{v} + \kappa\triangle\boldsymbol{\chi}, \tag{8}$$

$$\frac{\partial \omega}{\partial t} + \boldsymbol{v}\cdot\nabla\omega = -\frac{\partial T}{\partial x} + \nu\triangle\omega.$$

T 渦度の従う式 (9) は、3 次元ナビエ・ストークス系 (以下 3DNS と略す) の渦度の従う式と同等である。このことから、T 渦度も乱流ダイナミックスにおいて類似の役割を果たすことが期待される。実際、3DNS 系乱流で観察される微細秩序渦構造に似た、微細秩序 T 渦構造と呼べる T 渦層が形成される。その幅はおおよそコルモゴロフ長の 10 倍程度、長さがエネルギー長程度と幾何学的な特徴も似ている。更に、バーガース渦層と類似のバーガース T 渦層解が存在し、微細秩序 T 渦構造の第ゼロ近似となっている。

平均温度勾配が無く、低波数側で温度揺らぎを供給することで維持される一様乱流では、エントロピー ($S = T^2/2$) が高波数側にカスケードされる。ここで、エントロピー散逸率 ε_ϑ を以下のように定義する。

$$\varepsilon_\vartheta = -\frac{\mathrm{d}S}{\mathrm{d}t} = 2\kappa\int|\boldsymbol{\chi}|^2\mathrm{d}S = 2\kappa R.$$

コロモゴロフ流の仮説を用いると、非粘性極限でもエントロピー散逸率が有限であることが要請される：

$$\varepsilon_\theta = 2\kappa R < \infty.$$

ちなみに、慣性領域では、エントロピーとエネルギースペクトルは以下の冪則に従う。

$$S(k) = C_S k^{-7/5},$$
$$E(k) = C_E k^{-11/5}.$$

a 更に、2DFC 乱流でも間欠性が存在し、間欠性指数と微細秩序 T 渦構造が密接に関連していることが調べられている。

以上、2DFC 系と 3DNS 系の乱流特性は非常に似ていることを述べた。ところで、Pumir と Siggia は [25]、旋回流の局所近似としてオイラー方程式から非粘性２次元ブシネ近似方程式が直接導けることを示した。実は、彼らは以下で述べるように、３次元オイラー方程式の有限時間発散可能性を調べる目的で、非粘性２次元ブシネ近似方程式を扱っている。

5.2　非粘性非粘性ブシネ方程式の有限時間発散可能性

5.2.1　有限時間発散を調べる理由

３次元ナビエ・ストークス方程式及びオイラー方程式の有限時間発散可能性は乱流の基礎的理解や数学的な興味から詳細に調べられて来たが、今だ結論は得られていない。我々は、3DNS 乱流と類似の性質を持つ 2DFC 乱流が従う基礎方程式の非粘性形である２次元非粘性ブシネ方程式の有限時間発散可能性を新しい数値計算法 (AMR) を用いて調べているが、その存在の可能性を強く示唆する。

基礎方程式は、以下のように式 (6)、(7) で粘性項を除いた非粘性ブシネ近似方程式である：

$$\frac{\partial T}{\partial t} + \boldsymbol{v}\cdot\nabla T = 0$$
$$\frac{\partial \boldsymbol{v}}{\partial t} + \boldsymbol{v}\cdot\nabla \boldsymbol{v} = -\nabla p - \alpha g T \boldsymbol{e}_y.$$

我々が、非粘性 2DFC 方程式の特異性に興味を持つのは以下の３つの理由からである。(1) K41 の理論が非粘性極限 ($\nu \to 0$) で発散を示唆している、(2) 間欠性は特異性を基礎としており、有限時間発散解と何らかの関連がある、(3) 数学的な興味[*4]。

これらについては、前章で述べたので繰り返さない。3D オイラー方程式と非粘性 2DFC 方程式の間接的な類似性の他に、Pumir と Siggia により直接的な関係が導かれている。彼らは非粘性３次元軸対称旋回流が局所的に非粘性 2DFC 方程式で近似できることを示した。浮力が突然現れ不思議な感じがするが、これは旋回に起因する遠心力が見かけ上浮力と同じ効果を持つことによる。彼らは、非粘性 2DFC 方程式の有限時間発散を調べその可能性を示唆する結果を得た。しかし、彼らの用いたスキームは、方程式のスケール普遍性に基づく座標変換を用い高々256^2 の不等間隔メッシュを用いたものであり、その精度には疑問が持たれているため、結果に対する信頼度は高くない。

5.2.2　発散解のスケーリング

３次元オイラー方程式の有限時間発散に関して、その存在を仮定するならば ($t = t_0$；発散時刻)、以下の有名な不等式が成立する (Beale, Kato & Majda [1])。

$$\int_0^{t_c} |\omega|_{max} dt = +\infty.$$

[*4] CMI Millennium problem, http://www.claymath.org/

非粘性 2DFC 方程式では、これに加え、以下の温度勾配に関する不等式も同時に成り立つ（E & Shu [11]）。

$$\int_0^{t_c}\int_0^t \left|\frac{\partial T}{\partial x}\right|_{max} dsdt = +\infty.$$

渦度と温度勾配の最大値に関して以下のような冪的な発散を仮定すると、両不等式から指数に対し以下の不等式が得られる。

$$|\omega|_{max} \sim (t_c - t)^{-\alpha}, \quad \alpha \geq 1, \tag{9}$$

$$\left|\frac{\partial T}{\partial x}\right|_{max} \sim (t_c - t)^{-\beta}, \quad \beta \geq 2. \tag{10}$$

もし発散が相似的なら、もっとも単純な $\alpha = 1, \beta = 2$ が予想される。

5.2.3 これまでの計算との比較

3 次元 NS 方程式やオイラー方程式の発散解に関する研究は多数あるが、以下に関連のあるものについて表 1 にまとめた。 Kerr の仕事は、反対称な渦対を初期条件に用いた点で特徴がある [18]。

	初期条件	空間スキーム	格子	最大格子数	結果
Kerr	反対称渦対	Chebyshev	非一様	192	巾発散
Brachet et al	TG flow	Fourier	一様	864	指数増大
Grauer et al	渦管	AMR	非一様	2048	巾発散

表 1: 3 次元オイラー方程式の有限時間発散に関する、過去の仕事の一覧。格子数は、1 次元方向の局所的なメッシュ 数を表す。結果は一致していない。

このため、固定した不等間隔メッシュ（チェビシェフコロケーション点に対応）を用いることが可能となり、比較的少いメッシュでも高精度で特異性の存在が示唆される。特異性の研究に於いて、物理的考察が重要であることを示す例である。

Brachet 等は、テイラー・グリーン渦と言う高い対称性をもつ初期条件をスペクトル法を用いて調べた [5]。結果は、高々指数的増大の段階までしか追えなかった。この過程では、極めて薄い渦層が形成されるが、彼らはその不安定化が冪発散を起こすと述べている。

Grauer 等は、以下で説明する AMR 法を用いて発散があることを示唆する結果を得た [13]。ただし、空間構造には特に特異性の出現を示唆するような特異的な振舞は見られず、決定的ではない。また、時間発展にリワインドの手法を用いておらず、精度が低いため数段のレベルの積み上げしか行われていないように思われる。

非粘性 2DFC 方程式の特異性の探索は 3D オイラー方程式に比べ少い。これは、先にも述べた Pumir と Siggia の仕事 [25] に続く E と Shu の仕事 [11] で、存在が否定されたと考える研究者が多いことによると思われる。

非常に興味深いのは、Pumir と Siggia は特異性の出現が Brachet 等と同様に、形成された T-渦層の不安定性によると考えた。しかし、E と Shu は不安定化が起きないと結論付けている。 我々

の結果は、逆に不安定化が起こることを示唆しており、解析的な評価が待たれる。いずれにせよ、数値的に存在を示唆するためには、十分な空間解像度が必要である。

	初期条件	空間スキーム	格子	最大格子数	結果
P&S	バブル	相似変換	非一様	256	巾発散
E & Shu	バブル	Fourier	一様	1500	指数増大
我々	層	AMR	非一様	2^{21}	巾発散

表 2: 二次元自由対流問題での、過去の仕事と我々の仕事の比較。

5.2.4　数値計算スキーム

我々は、高精度の空間解像度を得るために、AMR （AMR）法を用いた。加えて、時間発展に対し精度を高めるため、リワインド（rewind ；巻戻し）と呼ぶ手法を用いた。以下、簡単に手法を紹介する。

AMR 法で必要となる誤差評価関数には、温度の 2 乗を用いた。これは、温度の任意関数はラグランジェの意味で保存量であるが、スキームはこの量の保存は保証しないからである。

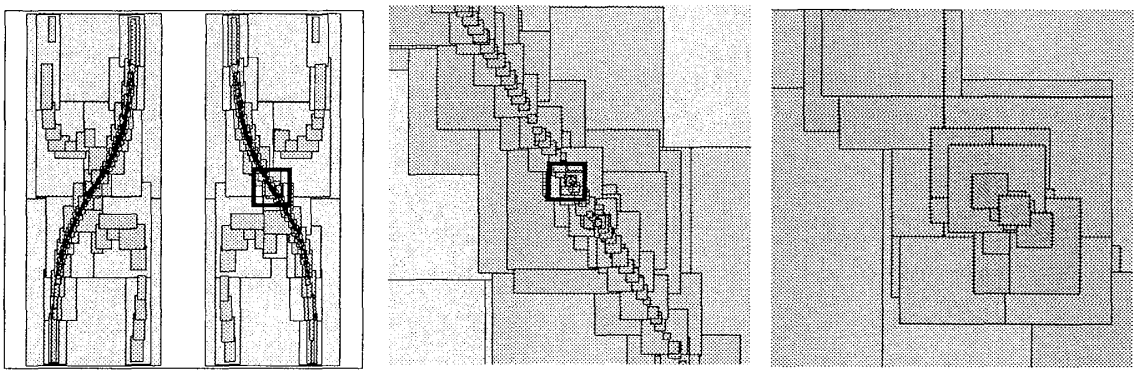

図 14: レベル生成の一例：15 段のレベルを積んだ状態。図中の太線で囲われた領域を拡大して表示している。強いフロントが形成されている所に細かいメッシュが集中している。

各格子での時間発展スキームは、空間方向の差分に TVD 条件を満たす 2 次の MUSCLE 法 [36] を、また時間発展には 2 次のルンゲ・クッタ法を用いた。ポアソン方程式の解法には、2 次精度の SCG 法を用いている。空間差分の精度を一致させるためには、高次差分を用いる必要がある。しかし、4 次精度で計算を再試したが、定性的な違いは見られなかった。また、高いレベルの情報が低いレベルにフィードバックされる効果も採り入れる必要があるかもしれないが、今回は無視した。境界条件は x、y 方向とも周期 2π の周期境界条件を用いた。重力は、y 軸方向下向きに取っている。

5.3 結果

この節では、これまで数値計算で得られた結果を紹介する。閾値の異なるいくつかのケースを行った。今回主に報告するケースはほぼ、0.4%の誤差内にありこの意味で十分精度があると言える。

5.3.1 スケーリング

図 15 に、$|\nabla T|$ と ω の最大値の時間発展を示した。図 15 は、$2 < t < 4$ の範囲で指数関数的増大が起きていることを示している。その後、増大は加速される。図 15 では、前章で紹介した、非粘性 2DFC 方程式の発散を仮定したときのスケーリング (9)、(10) を用いてフィットした。結果は、発散時刻 t_0 の選び方に依存する。しかし、増大は少くとも 4 重の指数関数より速いことは確認している。従って、この結果は有限時間発散を示唆していると見做せるであろう。

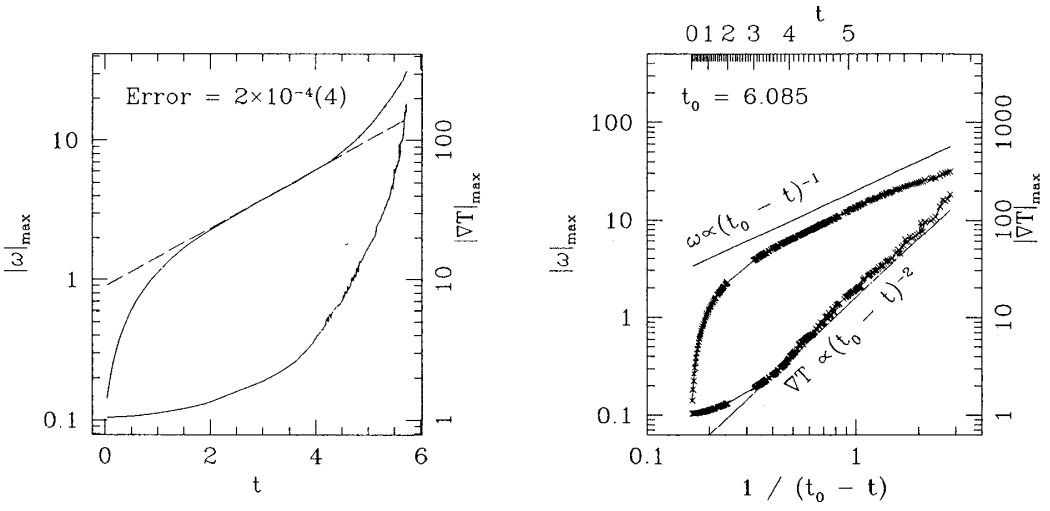

図 15: 渦度の最大値と温度勾配の最大値の時間変化。log-linear プロット (左) と逆冪関数によるフィッティング (右)。log-linear で、破線で直線を引いているが、k の直線に乗れば指数増大を示す。我々の計算結果は、明らかにこの直線からそれている。逆べき関数では、式 (9) および (10) で相似性が成り立つ場合の、$\alpha = 1, \beta = 2$ の直線を、比較のために示した。我々の結果は、この直線にほぼ近いものである。

初期発展では指数関数的増大が起きているが、これは以下に示すように、T 渦層の発達と対応する。このことは、空間解像度が低いと冪発散は観測されないことを示している。逆冪でのフィッティングは、発展の最終段階で合わなくなるが、シミュレーションの精度の影響かあるいは実際の現象を反映しているのかはわからない。ポアソン方程式の解法の精度を上げると、この傾向は強まるため、この現象は実際の現象を反映していると考えられる。

5.3.2　空間構造の発展

初期条件には、以下のような中央が高温で、その両脇に温度フロントがある状態を用いた。

$x \leq \pi$ の時：

$$x_c = \pi/2 - a\sin(y),$$
$$\omega = 0.1\exp\{-0.5 * [(x - xc)/w]^2\},$$
$$T = 0.5\{1.0 + \mathrm{erf}[(x - x_c)/\sqrt{2}w]\}.$$

$x \geq \pi$ の時：

$$x_c = 3\pi/2 + a\sin(y),$$
$$\omega = -0.1\exp\{-0.5 * [(x - x_c)/w]^2\},$$
$$T = 0.5\{1.0 + \mathrm{erf}[(x_c - x)/\sqrt{2}w]\}.$$

ここで、a はフロントの三角関数的変動の振幅、w はフロントの幅を表すパラメターで、それぞれ 0.3、0.4 とした。また、$\mathrm{erf}(x) = \dfrac{2a}{\sqrt{\pi}}\displaystyle\int_0^x \exp(-\xi^2)d\xi$ である。渦度は、バーガース T 渦層解の関係を満たすように決めた。

図 16(左) に初期の T 渦度の絶対値の等高図を示す。幅の広い T 渦層が対称に置かれている。図 16(右) には、時刻 $t = 4$ での発展を示した。T 渦層はよりシャープになっている。この発展は指数関数的増大ステージであり、3 次元オイラー方程式の場合の渦層の形成と対応していると考えられる。

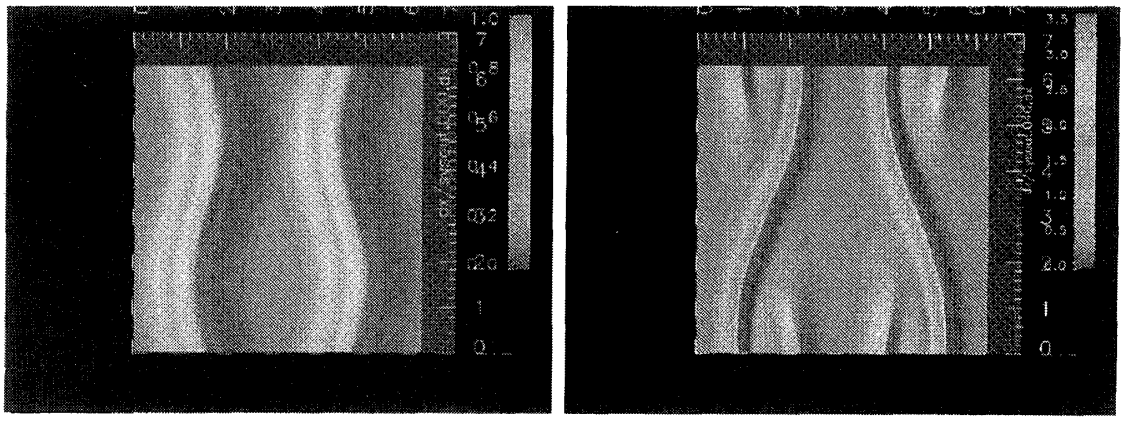

図 16: この問題の概念図。$|\nabla T|$ の分布を示す。t=0(左) および t=4(右)。

図 17(左) に、時刻 $t = 5.9$ での発展を示す。T 渦層は更に発展するが、その強さは渦層に沿って一様でない。図 17(右) に強 T 渦領域を拡大して示した。明らかに T 渦層は分裂しており、前段階の指数関数的な発展で形成された T 渦層が不安定化し、さらに非線形発展して局所的に分裂した構造に発達するように思われる。この実空間での T 渦層の不安定化以降の発展が、巾発散ステージに対応している。

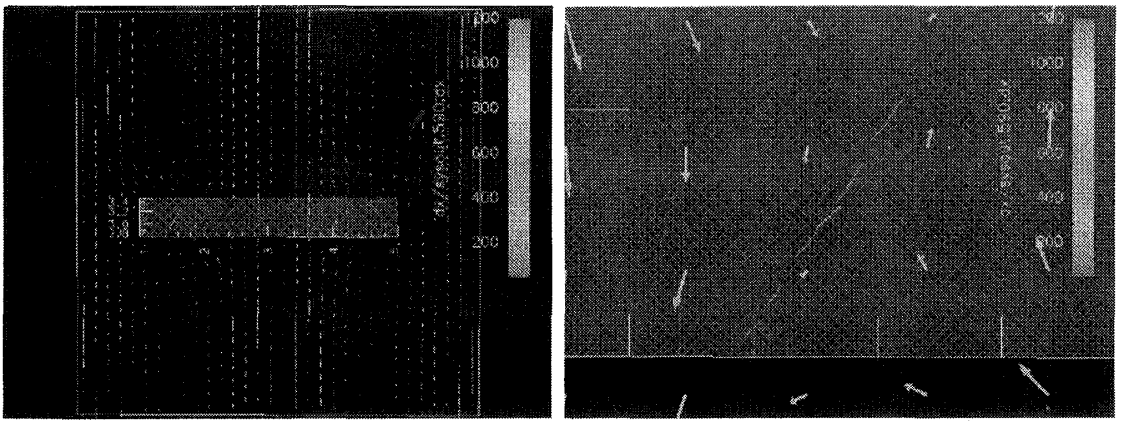

図 17: $|\nabla T|$: t=5.9 の全体図 (左) と、左側の停留点付近の拡大図 (右)。極大を示す等高線が、一つではなくいくつかの小さい極大に分裂している様子が分かる。引き延ばされ、分裂する課程を繰り返しながら、徐々に温度勾配の値を大きくし、発散に近づいているというシナリオを示唆するものである。

図 18(左) と 19(左) は、$t = 5.82$ と $t = 5.90$ の T 渦度の絶対値の等高線を示し比較した。これらは、図 17(左) の最大値の近傍を更に拡大したものである。図 17(左) では、6 本の線分として強 T 渦領域が判別できる。しかし、上から 3 本目の線分を拡大した図 19(左) でも、同様な T 渦が非一様に分布し、複数の線分から形成されていることが分かる。図 18(左) は少し前の状態を示しているが、まだ分裂は顕著ではない。従って、T 渦層の不安定化はスケールを小さくしながら繰り返し起き、全体として自己相似的なフラクタル構造を形成しているようにも見える。従って、巾発散は単純な過程では説明が出来ないと思われる。

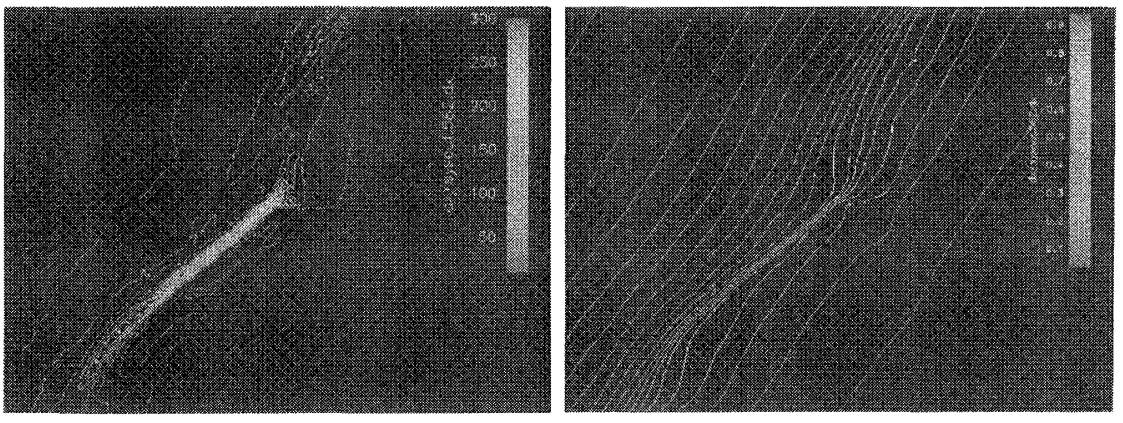

図 18: t=5.82 での様子の拡大図。温度勾配 $|\nabla T|$(左) と温度 T(右) の分布を示す。

図 18(右) と 19(右) に、$t = 5.82$ と $t = 5.90$ での温度の等高線を描いた。特異性は空間 1 点で発現するように思われる。このとき、温度勾配がこの点で発散するようである。実際、等高線が 1 点に集中しつつある。Constantin 等は、2 次元表面準地衡流 (SQG) での有限時間発散の可能性を調べたが、等温線がサドル状態の場合有限時間発散は無いことを示した [8]。しかし、我々の場合は、等温線にはサドルはなく単純なフロントであり、彼らの議論は適用できない。

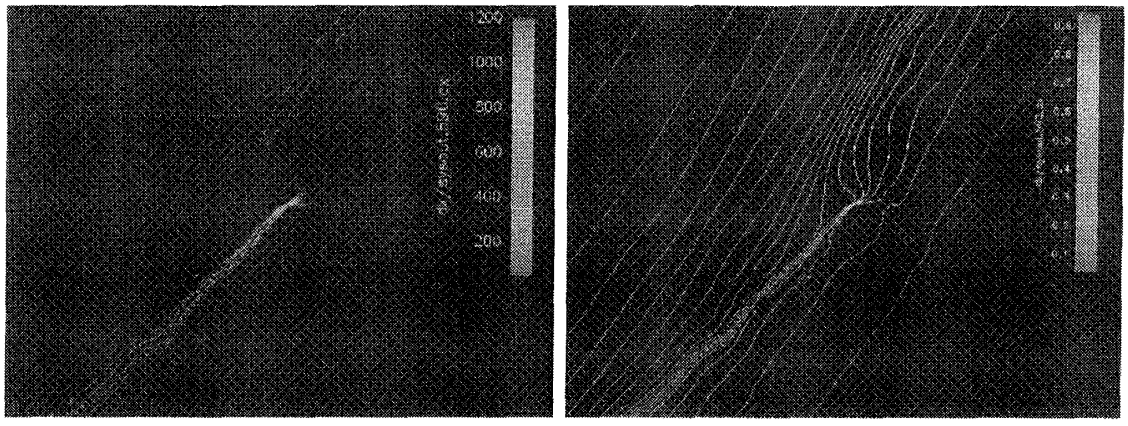

図 19: t=5.90 での様子の拡大図。$|\nabla T|$(左) と T(右)

5.4　まとめと課題

2 次元非粘性ブシネ近似方程式の有限時間発散の可能性を探るため、AMR 法を用いた高空間解像度のシュミレーションを行った。結果は、T 渦と渦度の最大値が巾的に発散することを示唆し、また有限時間発散の可能性を支持する。発展は、2 つのステージに分けられる。前段階では、T 渦層の発達に対応し、諸量は指数関数的に増大する。更に次の段階では、発達した T 渦層が不安定化し巾的な発散が起こる。この不安定化過程は自己相似的により小さなスケールで繰り返し起こり、T 渦層のフラクタル化を導くように思われる。しかし、フラクタル化は、まだ確認されておらず仮定にすぎない。特異性の発現は、複素空間での特異性が実空間に現れることと見做す研究者もいる。一般に非解析的な方程式に従う解は、複素空間に自然境界を持ち、またその構造がフラクタル的な自己相似性をもつことが知られているが、我々の場合の自己相似的な不安定化過程と関連があるかもしれない。

T 渦層の不安定化が重要な役割を果たしているが、この不安定化が初期に含まれる撹乱によるものか、あるいはスキームに由来する数値的な誤差なのかは現状では区別できない。今後、不安定化過程を詳細に調べることから、発散の物理的な機構を明らかにしていく予定である。本研究で用いたスキームは高精度で強力であるが、本文中でも述べたようにポアッソン方程式の解法に若干の不安がある。上部レベルの影響を下部にフィードバックさせる改良が必要かも知れないが、これについては今後の課題としたい。

非粘性での特異性が存在するとして、乱流状態を理解する上で重要かという疑問が常に問われる。我々は、少くとも間欠性の理解には有効であろうと考えている。また、高レイノルズ数で 3 次元微細秩序渦構造が崩壊するという報告もあるが、我々の場合も粘性が十分小さいならば、粘性があっても不安定化が起こることが予想できる。これが、乱流特性へ反映される可能性は否定できない。いずれにせよ、詳細な研究が必要である。

6 まとめにかえて

我々は、このコード開発を通じて、従来の AMR 法の適用限界を明らかにし、独自の改良によって、より汎用性の高い AMR 法の実装を実現することができた。

流体力学への応用例では、問題が二次元であるため可視化が比較的容易である。それでも、広く使われている AVS や我々が使っている IBM の OpenDX では、等間隔格子の場合は比較的容易に描画できるが、我々のような複雑なデータ構造を持つ場合に、うまく可視化する方法は確立しているとはいえない。実際我々は、有限要素法の研究者が独自に作って公開している triangle という、点集合を三角形分割するソフトウエアを組み合わせて、さらにデータ形式変換部分は独自に書いている。

さらに、これが三次元問題になった場合、可視化をどのような形で実現すればよいのか、今のところ検討中である。三次元単体は四面体だが、点集合をうまく四面体分割できるのか、四面体分割した中でうまく物理量の表示ができるのかは、これからの課題である。論文などに投稿するための図は、鳥瞰図とか等高面図とか、あるいは切断面での二次元図になるであろうから、問題はない。しかし、動作性のチェックとか実際にどのような現象がおこっているかを解析するために、何か三次元可視化システムと組み合わせる必要があるのではないかと考えている。

シミュレーションを用いた物理学研究においては、海外では 10 人規模のプロジェクト的研究が多く行われている。そういった開発体制に対抗し、独自に世界レベルの成果を上げるためには、我々のプロジェクト的な研究スタイル取り入れる必要があると痛感している。このようなプロジェクト的研究スタイルでは、現代的なコード設計手法がいかに有効であり、そのようにして開発されたコードは非常に応用範囲が広いものに成長し得る要素を内包している。今後、我々のプロジェクトに多くの優秀な (物理学研究者として、およびコードを書くという両方の意味において) 研究者が参加し、より汎用性が高いシミュレーションコードを完成してゆくことができれば良いと思う。

謝辞

本原稿を書くきっかけとなった数理解析研究所でのセミナーを企画し、本原稿を書くことを依頼してくださった、編集委員の大木谷さん、流体力学の分野での応用を提案し、研究上のハードウエアなどの提供を行ってくださった、藤定義さんおよび京都大学理学部物理教室の流体グループのみなさんに感謝いたします。

付録 A 弱解と差分変数の選択に関する補足

簡単のため、(2) の一次元の形、

$$\frac{\partial u}{\partial t} + \frac{\partial f(u)}{\partial x} = 0 \tag{11}$$

の中で、弱解の本質を理解するため、この方程式の非線形のうちで最も簡単なものとして、今 $f(u) = u^2/2$ の場合

$$\frac{\partial u}{\partial t} + \frac{\partial}{\partial x}\left(\frac{u^2}{2}\right) = 0 \tag{12}$$

を例に考えよう。

一般に、式 (12) の形の方程式は双曲型偏微分方程式と呼ばれ、数学的には時間的に有限の範囲でしか解を持たない。ここで解というのは、$u(x,t)$ を x の関数と見て、適当な連続性を持った初期条件 $u(x,0)$ を与えた場合、ある t に対して適当な連続性を持つ x 関数の集合の一つとして $u(x,t)$ が得られるという意味である。

この形の方程式は、物理ではたくさん現れるのに、適当な初期条件で無限時間解を持たないと言うのは不思議に思うかもしれない。物理的には、この事情は次のように言い換えることができる。「ある時刻までこの方程式を積分すると、ある種の不連続が発生するために、解は連続ではなくなる。」つまり、ショックが発生するということである。そこで、不連続がある場合に、方程式が意味するものは何かということを考え、解の拡張を考えたものが、弱解である。

この方程式の特徴として、特性線が存在する。特性線とは、その線上で物理量が一定になるような線であり、これを式で表すと

$$\frac{du}{dt} = 0$$

である。微分の順序が変更できる程度に十分な連続性を持つ範囲では、式

$$\frac{dx}{dt} = \frac{\partial \phi}{\partial u}$$

が、特性線の満たすべき方程式ということになる。数学的に有限時間しか解が定義できないことを示すために、よく用いられるのは、

$$u(0,x) = -\mathrm{th}\left(\frac{x}{2\varepsilon}\right) \tag{13}$$

式 (12) の上の初期条件でである。初期条件は連続だが、この初期条件では、解は、$t < 2\varepsilon$ でのみ存在する。この事情は、特性線で書けば、$t < 2\varepsilon$ で特性線が交わってしまうことを意味している。

不連続が存在する場合、式 (11) は物理的にはどう解釈されるかを復習してみよう。この式は、u という物理量の保存則である。つまり、

$$\int_\Gamma u dx - f(u)dt = 0 \tag{14}$$

となる。微分可能な解が存在する領域では、この二つの方程式が同値であることが証明されている [16, 17]。解を区分的滑らかな関数にまで拡張した場合に、基本方程式は式 (11) の自然な拡張として、式 (14) が使われることになる。これから、不連続線の上では

$$\frac{dx}{dt} = \frac{f(u_1) - f(u_2)}{u_1 - u_2} \tag{15}$$

が成立している必要がある。ここで、u_1 と u_2 は、それぞれ不連続線の両側の値である。これは、たとえば流体のショックで言えば、ショックの両側の物理量がわかればショックの速度が導ける、いわゆる Rankine Hugoniot 関係に相当する。

さて、式 (12) は、十分な連続性を持つ範囲で、

$$\frac{\partial}{\partial t}\left(\frac{u^2}{2}\right)+\frac{\partial}{\partial x}\left(\frac{u^3}{3}\right)=0 \tag{16}$$

と変形できる。式 (12) と式 (16) について、それぞれ式 (15) はどういうショック速度を要求するのかを見てみる。式 (12) では、

$$\frac{dx}{dt}=\frac{u_1^2/2-u_2^2/2}{u_1-u_2}=\frac{u_1+u_2}{2}$$

となり、式 (16) では

$$\frac{dx}{dt}=\frac{u_1^3/3-u_2^3/3}{u_1^2/2-u_2^2/2}=\frac{2}{3}\frac{u_1^2+u_1u_2+u_2^2}{u_1+u_2}$$

となり、同じではない。これは、u が保存変数なのか、u^2 が保存変数なのかという違いに対応する。注意を要する点は、式 (12) と式 (16) の関係は、流体の基礎方程式においては、Euler 方程式での運動量の保存則を用いるのか、それを速度の時間微分の式に書き直した Navier-Stokes 方程式の粘性項を取り去った式を用いるのかの違いに他ならないことである。つまり、運動量保存則を用いるなら運動量が保存量である問題を解いていることになり、Navier-Stokes 方程式の粘性項を落とした式を用いるなら「速度という保存量」の存在を仮定して問題を設定していることになる。ショックが発生するような状況では、この二つの式は同じではないのである。なお、この二つは、微分方程式としては同じ方程式に見えるが、積分系の式 (14) の形に書いたときには異なる式である。

Lax と Wendroff は、適切な差分式、すなわちある精度で微分方程式が差分方程式になるように差分化された式 (11) の差分式は、不連続がある場合には、この拡張である式 (14) から導かれる弱解に収束することを証明した [22]。すなわち、不連続がある場合には、保存量を変数とした差分式を書かなければ、数値解は求める正しい解にならない。これは、Navier-Stokes 方程式を用いてはならないと言っているのではない。Navier-Stokes 方程式は、粘性の影響で完全な不連続に至らないので、粘性を含む場合に Navier-Stokes 方程式を使うことは誤りではない。

付録 B　計算法と TVD 性に関する補足

先ほどのモデル方程式 (11) を解く場合を考える。この方程式は、解が十分な連続性を持つ範囲では、特性線を持つ。特性線に沿って値は一定なので、積分

$$\int |u|dx$$

は時間によらず一定値になる。解が不連続を含む場合は、物理的にはショックと解釈される。前章で示したとおり、不連続をまたぐ物理量の間の関係は、弱解という解の拡張により規定される。しかし、元々不連続がある場合はよいが、解の途中で不連続が発生する場合はどうか。今、二つの初期条件

$$u_1(x)=1,\text{if } x\le 0, u_1(x)=-1,\text{if } x>0$$
$$u_2(x)=-1,\text{if } x\le 0, u_2(x)=1,\text{if } x>0$$

を考える。これらの場合、$(x,t)=(0,0)$ を通る直線が一つしかないという条件を課せば、解は一意に定まる。しかし、この点から 3 本の不連続線が出ているとすれば、それらの不連続線の間の値の選択の自由度により、解は無数に存在できる。また、初期条件 $u(0,x)=u_2(x)$ を採用した場合、$x-t=0$ と $x+t=0$ の間を連続的に変化するような解も考えられる。

このように、数学的には、不連続を含む場合の解は不定である。いったん特性線が交わると、その後にどのような不連続を生成して解が発展するかは、原理的に無限 (実数密度) の可能性がある。その中で、どのような解が物理的に意味ある解であるかを決めるのは、エントロピー増大則であることが知られている。

不連続をまたぐ流れに沿った流体粒子について、エントロピー増大則を課すと、積分

$$\int |u| dx$$

は増大することはないということがわかる。この条件を差分式に適用したもの、

$$TV(t_n) \equiv \sum_i |u(t_n)_{i+1} - u(t_n)_i|$$
$$TV(t_{n+1}) \leq TV(t_n) \cdots \leq TV(0)$$

が、TVD 条件 (Total Variation Diminishing condition) と呼ばれる。我々の数値計算でも、ベースになる差分スキームにはこの条件を課している。

実際、この条件を課さない差分スキームを適用すると、あちらこちらに密度極大が現れて、これらが独立に増大することになる。これは数値不安定であり、実際の物理的不安定ではない。

参考文献

[1] T. Beal, T. Kato, and A. Majda. *Commun. Math Phys.*, 94:61, 1984.

[2] J. J. Berger and J. Oliger. Adaptive mesh refinement for hyperbolic partial differential equations. *J. Comput. Phys.*, 53:484–512, 1984.

[3] M. J. Berger and P. Colella. Local adaptive mesh refinement for shock hydrodynamics. *J. Comput. Phys.*, 82:64–84, 1989.

[4] Marsha J. Berger and Randall J. Leveque. Adaptive mesh refinement using wave-propagation algorithms for hyperbolic systems. *SIAM. J. Numer. Anal*, 35(6):2298–2316, December 1998.

[5] M.E. et al Brachet. *Phys. Fluids*, A4:2845, 1929.

[6] J. M. Burgers. *Adv. Appl. Mech.*, 1:171, 1948.

[7] J.D. Cole. *Q. Appl. Math.*, 9:225, 1951.

[8] P. Constantin, Q. Nie, and Schörghoher. *Phys. Rev. E.*, 60:28583, 1999.

[9] J. Crank and P. Nicholson. A practical method for numerical evaluation of solutions of partial differential equations of heat-conduction type. *Proc. Cambridgs Philons. Soc.*, 43:50–67, 1947.

[10] Darren De Zeeuw and Kenneth G. Powell. An adaptively refined cartesian mesh solver for the euler equations. *J. Comput. Phys.*, 104:56–68, 1993.

[11] W. E and C.W. Shu. *Phys. Fluids*, 6:49, 1994.

[12] Holger Friedel, Rainer Grauer, and Christiane Marliani. Adaptive mesh refinement for singular current sheet in incompressible magnetohydrodynamic flows. *J. Comput. Phys.*, 134:190–198, 1997.

[13] R. et al Grauer. *Phys. Rev. Lett.*, 80:4177, 1998.

[14] Rainer Grauer, Christiane Marliani, and Kai Germaschewski. Adaptive mesh refinement for singular solutions of the incompressible euler equations. *Phys. Rev. Lett.*, 80(19):4177–4180, 1998.

[15] E. Hopf. *Comm. Pure Appl. Math.*, 3:201, 1950.

[16] O. A. Oleĭnic. Discontinuous solutions of non-linear differential equations. *Uspekhi Mat. Nauk*, 12:3–73, 1957.

[17] O. A. Oleĭnic. Discontinuous solutions of non-linear differential equations. *Amer. Math. Soc. Transl. Ser. 2*, 26:95–172, 1963.

[18] R. M. Kerr. *Phys. Fluids*, A5:1725, 1993.

[19] A. M. Khokhlov. Fully threaded tree algorithms for adaptive refinement fluid dynamics simulations. *J. Comput. Phys.*, 143:519–543, 1998.

[20] R. I. Klein, C. F. McKee, and P. Colella. On the hydrodynamic interaction of shock waves with interstellar clouds. *Astrophys. J.*, 420:213, 1994.

[21] Steven E. Koch and Jeffery T. McQueen. A survey of nested grid techniques and their potential for use within the mass weather prediction model. *NASA Technical Memorandum*, (87808), February 1987.

[22] Peter Lax and Burton Wendroff. System of conservation laws. *Comm. Pure and Applied Math.*, 13:217–237, 1960.

[23] J. E. Melton, M. J. Berger, M. J. Aftosmis, and M. D. Wong. 3d applications of a caratesian grid euler method. *AIAA Paper*, 95-0853, 1995.

[24] M.L. Norman and G.L. Bryan. Cosmological adaptive mesh refinement. astro-ph/9807121, July 1998.

[25] A. Pumir and E.D. Siggia. *Phys. Fluids*, A4:1472, 1992.

[26] J. J. Quirk. *An Adaptive Grid Algorithm for Computational Shock Hydrodynamics.* PhD thesis, Cranfield Institute of Technology, U.K., 1991.

[27] P.L. Roe. Approximate riemann solvers, parameter vectors and difference schemes. *J. Comput. Phys.*, 43:357–372, 1981.

[28] Gary A. Sod. A survey of several finite difference methods for systems of nonlinear hyperbolic conservation laws. *J. Comput. Phys.*, 27:1–31, 1978.

[29] S. Toh and T. Matsumoto. In T. Kambe, T. Nakano, and T. Miyauchi, editors, *IUTAM*

Symposium on Geometry and Statistics of Turbulence, page 279, 2000.

[30] S. Toh and E Suzuki. *Phys. Rev. Lett.*, 73:1501–1505, 1994.

[31] J. Kelly Truelove, Richard I. Klein, Christopher F. McKee, John H. Holliman II, Louis H. Howell, and Jeffrey A. Greenough. The jeans condition: a new constraint on spatial resolution in simulations of isothermal self-gravitational hydrodynamics. *Astrophys. J. Lett.*, 489:L179–L183, November 1997.

[32] Bram van Leer. Towards the ultimate conservative difference scheme, I. the quest of monotonicity. volume 18, pages 163–168, 1973.

[33] Bram van Leer. Towards the ultimate conservative difference scheme, II. monotonicity and conservation combined in a second-order scheme. *J. Comput. Phys.*, 14:361–370, 1974.

[34] Bram van Leer. Towards the ultimate conservative difference scheme, III. upstream-centered finite-difference schemes for ideal compressible flow. *J. Comput. Phys.*, 23:263–275, 1977.

[35] Bram van Leer. Towards the ultimate conservative difference scheme, IV. a new approach to numerical convection. *J. Comput. Phys.*, 23:276–299, 1977.

[36] Bram van Leer. Towards the ultimate conservation difference scheme V. a second-order sequel to godunov's method. *J. Comput. Phys.*, 32:101–136, 1979.